Decontamination from Black Virus Using Parallel Strategy

by

Yichao Lin

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the Master degree in
Computer Science

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

In this thesis, the problem of decontaminating networks from *black viruses* (BVs) using parallel strategy with a team of system mobile agents (the BVD problem) is studied. The BV is a harmful process whose initial location is unknown a priori. It destroys any agent arriving at the network site where it resides, and once triggered, it spreads to all the neighboring sites, i.e, its clones, thus increasing its presence in the network. In order to permanently remove any presence of the BV with as less execution time as possible and minimum number of site infections (and thus casualties), we propose parallel strategy to decontaminate the BVs: instead of exploring the network step by step we employ a group of agents who follow the same protocol to explore the network at the same time, thus dramatically reducing the time needed in the exploration phase and minimizing the casualties. Different protocols are proposed in meshes, tori, and chordal rings following the monotonicity principle. Then we analyze the cost of all our solutions and compare to the asynchronous BV decontamination. Finally conclusion marks are presented and future researches are proposed.

Acknowledgements

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

A distributed system is a group of computational entities cooperating with each to achieve one or more tasks. This thesis deals with distributed computing by mobile agents in network. More specifically, we deal with the problem of deploying a group of mobile agents who follow the same protocol to explore the network and decontaminate the dangerous virus (called Black Virus) present on the network nodes.

In this chapter, the motivations of the problem are provided, following is a brief summary of the contributions.Finally, an overview of the organization of the thesis is presented.

1.1 Problem and Motivation

Mobile agents are widely used in distributed and network systems while the applications of them can cause some security issues, thus threatening to the network: A contaminated or infected host can destroy working agents for various malicious purposes; A malicious agent can contaminate or infect other computer nodes so they become malfunctional or crash. The harmful hosts, often called *Black Holes* trigger the problem called *Black Hole Search* (BHS), the focus of which is to locate their positions since it is statics. This problem has been studied in many variants. For example, different topologies and different settings (synchronous and asynchronous). The harmful agents trigger the problem called *Intruder Capture* (IC). Its main focus is to deploy a group of mobile agents to capture a extraneous mobile agent (the intruder) who moves arbitrarily fast through the network and infects the visiting sites. Also it has been investigated in a variety of topologies. More detailed literature review will be provided in Chapter 2. Note that BH is static and only damage the agents reaching it without leaving any detectable trace. Intruder is mobile and harmful to the network nodes but does not cost any harm to other system agents. A new harmful presence called *black virus* BV has been initially introduced by Cai et al. in[**?**]. It is a dangerous process resides at an unknown site in a network and destroys any upcoming agents, but unlike the BH, the node where the original BV resides thus become clean. At the same time, the original BV multiplies (called clones) and spread to all neighbouring nodes, thus increasing its number,and damage in the network. A BV is destroyed when it moves to a site where there is already an agent. Based on this harmful presence, a new problem called Black Virus Decontamination(BVD) is presented by Cai et al., the main focus of which is to use a group of system agents to permanently remove any presence of the BV from the network. A protocol defining the actions of the agents solves the BVD problem if at least one agent survives and the network is free of BVs. Also, a desirable property of a decontamination protocol is that the nodes which have been explored or cleaned by mobile agents are not be recontaminated by the BV spreading. A solution protocol with such a property will be called *monotone*. see[**?**]. Some important cost measure is the number of node infections by the BVs (casualties); size of the team, i.e, the number of agents employed by the solution, the time needed by the solution. Solutions in which the agents explore the network's node in sequence have been proposed in [**?**], [**?**] and [**?**]. The size of team is minimum in [**?**] and the number of site infections is also minimum in such case,i.ie., exploring the network nodes in sequence. Now we are interested in the solution using parallel strategy where we deploy a larger number of mobile agents following the same protocol to decontaminate the network in the exploring phase with the goal to minimize the *total working time* (TWT) which is caculated by multiplying the number of agents and the total execution time and also the casualties.

1.2 Our Contribution

1. In this thesis, we propose parallel strategy to solve the BVD problem. It is the first attempt to deal with this issue in a parallel way. Agents are not allowed to communicate with each other unless they are in the same network node so the protocol should enable the agents in different nodes to move properly, i.e, the route of every agent is different but they are served to explore the network; when a BV is triggered, other agents should bypass the new-formed BVs... We give simple but efficient solution to deal with this problem with acceptable cost. Also we give the size of the exploring team which is minimum to guarantee both the TWT and the casualties we reach.

2. The BVD problem is investigated for three important topologies: *meshes*, *tori*, *chordal rings*. All the protocol are optimal both in term of TWT and casualties. We compare our solution with [**?**] and [**?**] in which the exploring route is in sequence and the result is that our solution is better than that of them in both TWT and casualties. One should be point out that in chordal ring especially, the more complicated the chordal ring becomes, the more TWT that we save comparing to [**?**].

1.3 Thesis Organization

In this section, we will present the organization of the thesis. After this chapter we review literature on topics related to our problem. We begin by reviewing some papers that address the black hole search problem in different settings, variations and topologies. We then review some papers concerning the decontamination problem, sometimes referred to as intruder capture, in different settings and variations. Finally, we review the only paper that investigates black virus disinfection in several topologies.

In chapter 3 we introduce the problem in question and provide definitions and terminology regarding black virus disinfection. We also present our assumptions and topology (the chordal ring) and describe the high level ideas that serve as the basis for all of our solutions.

In chapter 4 we go through our solution for double loop chordal rings. We describe in detail the two phases and their complexities. For the second phase, we introduce three deployment strategies: *Move-optimal*, *Simple greedy* and *Smart greedy*. For the move-optimal deployment strategy, we present the solution for two cases: shortly-chorded double loops (where the non-ring chord is small compared to the size of the ring) and the general double-loop. Our bound on the move-complexity is tight for shortly chorded double loops.

In chapter 5 we address the problem in triple loop chordal rings. We describe in detail the two phases and their complexities. For the second phase, we describe only the move-optimal strategy since the greedy approach does not always work for this type of chord structure. For simplicity, we only consider the shortly-chorded triple loops in the calculation of the upper bound on the number of moves. We also show that the bound is tight in two special extreme cases.

In chapter 6 we address the problem in chordal rings with consecutive chords. For the second phase, we describe only a local greedy strategy (One-direction greedy) because it also provides an optimal-move solution for this particular chordal ring.

In chapter 7 we discuss the problem in arbitrary chordal ring structures. The surrounding solution that we propose is very general and works with any chord structure, however, it is not optimal. We provide upper bounds to the path lengths.

In chapter 8 we conclude our thesis by summarizing all the results and discussing some open problems related to the *black virus* disinfection topic.

Chapter 2

Related Work

In this chapter we shed light on several topics related to black virus chordal ring disinfection by reviewing the literature related to our problem. Many different problems have been studied using the mobile agent model in the field of distributed computing. *Black hole search* and *Decontamination* are two examples of the ways in which mobile agents are deployed to find solutions from a theoretical point of view. The rest of this chapter is divided into the following sections: black hole search, decontamination and black virus disinfection.

2.1 Black Hole Search

The *Black Hole* (BH) has been extensively studied over the past decade. It was initially introduced by [15] as a faulty node that has a destructive impact on any visiting mobile agent. This hostile node is static and its location is unknown a priori; moreover, after destroying an agent, it leaves no noticeable trace. It is for this reason that locating it is problematic. The *Black Hole Search* (BHS) problem consists of having a team of mobile agents explore a connected network that contains a BH in order to locate it. The search is terminated once the BH is found and at least one agent survives. The primary issue regarding this problem is locating the hostile node within reasonable complexity. Complexity is usually measured in terms of the size of the team of agents, the number of movements and time. The static nature of the *black hole* makes it destructive only to visiting mobile agents and not to other nodes in the network. This problem has been widely discussed in different settings and variations. The following settings and variations will be discussed below: synchronicity, topological characteristics, agents capabilities and number of black holes.

2.1.1 Synchronicity

Synchronicity refers to the execution timing of agent movements and computations. The timing can be *Synchronous* or *Asynchronous*. When we have synchronous agents, the agents consume one unit of time to traverse a link connecting two neighbouring nodes while computation time is negligible. On the other hand, when we have asynchronous agents, their movements and computations consume a finite but unpredictable amount of time. [28] demonstrates that the execution timing has an impact on overall complexity. In this paper, they assume that G is a directed graph , n is known and that Δ represents the incident edges to the BH.They demonstrate that in the case of synchronous execution, $size = O(\Delta.2^{\Delta})$. In regards to the lower bound for the number of required agents, it has been demonstrated that two agents suffice when $\Delta = 1$ and four agents when $(\Delta = 2)$. However, in the asynchronous case, when $(\Delta = 2)$ a minimum of five agents are needed. In [12], the authors investigate asynchronous execution in several common topologies. This paper demonstrates that when agents have complete topological knowledge, $size = 2$ and the number of moves is $\Theta(n)$.

2.1.2 Topological Characteristics

As previously mentioned, black hole locating has been investigated in a wide variety of settings. One important aspect of the black hole search that affects the solution is the nature of the topology. The topology may be directed or undirected, arbitrary or common and have one or more homebases.

Directed or Undirected. Whether the topology is directed or not has an impact on the difficulty of finding a solution for the BHS problem. Some studies (e.g. [11, 13, 14, 15]) have suggested the use of the *cautious walk* technique when exploring a graph in order to minimize the number of agent casualties. In the *cautious walk* technique, an agent traverses a link back and forth to mark it as a safe edge so that other agents can traverse it afterwards. This technique has to be modified when the graph is directed. In the case of a directed graph, different approaches have been suggested (e.g. [10, 15, 28]). Complexity changes significantly depending on whether the graph is directed or not. For instance, when Δ represents the incident edges to the BH, if an arbitrary graph is undirected, $size = \Delta+1$ ([14]) . If the graph is directed, $size \geq 2^{\Delta}$ ([10]). This also applies to common topologies such as rings ([15]). [15] demonstrates that in specific settings where the ring is directed, we need $(\Theta n \log n)$ number of moves. The complexity is almost doubled if the ring is undirected.

One Homebase or Several. Mobile agents begin their mission from one or more nodes called *homebases*. Some studies (e.g. [10, 11, 12, 14, 15, 28]) rest on the assumption that all agents start from a single *homebase*. In this case, agents are referred to as 'co-located' agents. Other studies (e.g. [7, 14, 21, 22]) rest on the assumption that there are several *homebases*. Agents in this case are referred to as 'dispersed' or 'scattered' agents. The number of homebases significantly affects the strategy and overall complexity. In [7], the authors solve the BHS problem in a synchronous ring in two variants, directed and undirected, using dispersed agents. This paper shows the optimal number of dispersed agents with specific resources when they use the 'pure token' communication method. In order to explore an unsafe graph, the authors of [21] propose an algorithm that solves the exploration problem in the presence of faulty nodes and links using asynchronous dispersed agents who can communicate via the whiteboards. In the worst case scenario, the algorithm proposed in this study requires $O(n_s m)$, where m represents the number of edges and n_s represents the number of safe nodes in the graph.

2.1.3 Agent Capabilities

The various capabilities granted to agents have an impact on solving the BHS problem. Some of these capabilities are discussed in the following section:

Communication Mechanisms. In the mobile agents model, agents communicate with each other in different ways in order to infer the location of the BH. In the literature, three essential communication methods have been studied: whiteboard, tokens and time-out. Each mechanism has an impact on the proposed solution. In the whiteboard model there is a storage space located at each node and agents are able to read and write there (see [10, 14, 21, 28]). In the token model, agents use tokens which act like memos. These tokens can be dropped off and picked by agents at nodes or edges (see [7, 20]). The time-out mechanism can only be used using synchronous execution where agents have a pre-determined amount of time to perform a task (see [8, 9, 11]).

Topological Knowledge. The degree of knowledge the agents have regarding the topology determines how solutions are designed. In some cases, a minimum extent of knowledge is mandatory. In asynchronous execution, without the knowledge of n, the total number of nodes, the BHS problem can not be solved because any solution without knowledge of n will never terminate. [14] studies the ways in which the extent of agent knowledge changes complexity. In this paper, Dobrev et al. cover three different types of topological knowledge in an asynchronous arbitrary network. If agents are only aware of the total amount of nodes and have no knowledge of the graph structure, $size = \delta + 1$, where δ represents the highest degree of a node and $size$ is the number of agents, and $\Theta(n^2)$ number of moves. If the agents have no knowledge of the total amount of nodes but have a sense of direction, $size = 2$, and $\Theta(n^2)$ number of moves. If agents have complete topological knowledge, $size = 2$, and $\Theta(n \log n)$ number of moves.

2.1.4 Number of BHs

The black hole problem has been investigated in two scenarios: single and multiple BH. Searching for a single BH is less destructive. The majority of the studies mentioned earlier focus on locating a single BH. Some studies consider multiple BHs (see [22, 27]). In [27], Kosowski et al. demonstrate that in a directed graph, to solve the BHS problem, n and Δ should be known to the agents, and the solution would cost at most $O(\Delta.2^{\Delta})$ synchronized agents.

2.2 Decontamination

Because of network connectivity, a moving object can contaminate an entire network, resulting in dysfunctional performance. This scenario is considered to be one of the most critical threads in network security (ie., viruses and how they attack devices on the Internet). As a result, there is a need for a cleaning and protecting process (*Decontamination*). This problem has been extensively studied in graph theory under several names: graph search, intruder capture and decontamination. This problem was first introduced in 1967 by [4]. The decontamination process can be conducted internally or externally. Internal decontamination allows a node to perform the task without external help. In other words, the nodes are able to decontaminate themselves locally. For example, running an Anti-virus program on an infected device could rid that device from viruses it got from the Internet. On the other hand, external decontamination requires external intervention, such as mobile agents. An agent decontaminates infected nodes by passing through them, yet the nodes are vulnerable to recontamination once the agents depart.

The Mobile Agents Model is a form of external decontamination in which moving agents are used to disinfect a contaminated network. The mobile agents must decontaminate the network in a way that prevents recontamination. In this model, nodes have three states: contaminated, decontaminated or guarded. A node is decontaminated once an agent passes through it, guarded once an agent resides in it and otherwise contaminated. Initially, all nodes are contaminated with the exception of the homebase which is guarded. This model has many variations and parameters that have a significant impact on the overall complexity. As a result, there are many protocols and strategies that handle this problem in various topologies. These protocols were designed to be *monotone*, that is, once a node is decontaminated, it can not be re-infected.

Usually, the complexity of such protocols is measured by the number of agents used in the decontamination process, the number of moves the agents make and the amount of time needed to decontaminate an entire network.

The following variations have a significant impact on protocol performance and complexity: synchronicity, agent capabilities and topological characteristics.

2.2.1 Synchronicity

For synchronous execution many papers address the problem of decontamination where agents synchronize their movements in such a way that the explored nodes never get re-contaminated (see [3, 16, 29]). In the case of asynchronous execution, there is a need for a coordinator who organizes other agents, ensuring that all agents are in the correct positions before starting a new step (see [18, 19, 23, 24, 29]).

2.2.2 Agent Capabilities

In this problem, agents have been given certain capabilities including cloning, visibility and immunity.

Visibility. Some studies have granted agents the visibility, which is the ability to see whether the neighbouring nodes are clean or contaminated (see [18, 19]). Some studies have only employed the local model in which agents only know the information (state, ports) for the node in which they reside ([18, 19, 23]). Visibility impacts complexity since the agents are able to move in an autonomous fashion, without the need for coordination. In [18], Flocchini et al. demonstrate that to decontaminate a chordal ring $C_n\{d_1 = 1, d_2, ..., d_k\}$ in the local model, the number of agents required is $(2d_k + 1)$, while in the visibility model, the number of agents required is $(2d_k)$.

In [19], Flocchini et al. compare the complexity of decontaminating a hypercube in both models. In the local model, they propose an algorithm that requires $\Theta\ (\frac{n}{\sqrt{logn}})$ number of agents and $O(n \log n)$ moves. In the visibility model, they propose an algorithm that requires $(\frac{n}{2})$ agents and $O(n \log n)$ moves.

Cloning. Cloning refers to an agent's ability to make copies of itself. This feature would reduce the number of moves but would not produce an optimal number of agents. In [19], Flocchini et al. study the impact of the cloning feature on the decontamination of a hypercube. They demonstrate that using cloning, the number of moves is drastically reduced: $(n - 1)$ in both the visibility and local models.

Immunity. Immunity means that a node is immune from recontamination after an agent departs. In the literature, there are two types of immunity: local and temporal.

In local immunity, the immunity of a node is determined by the state of its neighbours. For instance, a node stays clean after the departure of an agent unless the majority of its neighbours are contaminated. If the majority of its neighbours are contaminated, that node gets re-contaminated. In other words, the immunity level in this model is about half of the node's degree. If the contamination reaches more than half, the node gets re-contaminated (see [16, 29]). In temporal immunity, a node is immune for a specified amount of time (t). After the departure of an agent, the node is clean for t time regardless of the state of its neighbours. After the time elapses, the node is vulnerable to recontamination if it has at least one contaminated neighbour (see [24]). The two types of immunity provide a network with some resistance to recontamination. In the absense of these two models, the immunity of the nodes is nil, meaning that the node can be recontaminated if any of its neighbours are contaminated.

2.2.3 Topologies

The mobile agent decontamination model has been studied in various topologies. In this section we list the results of decontamination protocols in some common topologies.

Tree. Many researchers have studied the decontamination process in trees. The basic strategy for decontaminating a tree is to divide the tree into sub-trees starting from a single homebase (the root), see [24, 29].

The best case scenario in terms of the number of agents required occurs when the homebase is the first or last node and we have a line that requires one agent for decontamination. If the homebase is any other node, we require two agents. The worst possible case is the decontamination of a complete binary tree.

In [29], Luccio et al. consider an asynchronous environment, continuous search, visibility mode, local immunity, single homebase, and a monotone strategy. In order to enforce local immunity there are three basic rules controlling agent departure from a node. If the number of clean neighbours is lower than half of the total number of neighbours (i.e. the majority), the agent does not depart. If the number of clean neighbours is equal to half of the total number of neighbours, the agent only moves to a contaminated neighbour. If the number of clean neighbours is greater than half of the total number of neighbours, the agent moves to any neighbour. To identify the minimum number of cleaning agents, this paper divides trees into two classes: trees with degree $d \leq 3$ (e.g. binary tree, line) and general arbitrary trees.

- Trees with degree $d \leq 3$ can be decontaminated using a maximum of two agents, while the number of moves is $(2(n-1)-diam)$, where $diam$ represents the diameter.
- For general arbitrary trees with degree $d > 3$, this paper introduces the state of stability. A node x is stable if it satisfies the following two conditions: x is immune to recontamination whether it is guarded or not, and the majority of its children are stable. The authors use these conditions to determine the minimum number of cleaners needed to keep a node stable. As a result, we can determine the minimum number of agents required to decontaminate the entire tree. Any lower bound on the minimum number of agents needed to stabilize x provides a lower bound on the minimum number of agents needed to decontaminate a tree rooted in the homebase. In this protocol, the agent stabilizes the root and then recursively moves to unstable nodes and stabilizes them until all nodes are stable. The number of moves in this protocol is not proven. This study demonstrates how local immunity improves complexity (minimum number of cleaners), especially in a binary tree.

In [24], the authors consider a synchronous environment, continuous search, temporal immunity (i.e. $t > 0$), single homebase and a monotone strategy. In this case, any node is immune from recontamination when it is guarded and during the immunity time t after the agent leaves. Therefore, agents can move back and forth during period t. In other words, an agent can leave a node and traverse a path of length up to t/2 and go back to that node without worrying about recontamination. Using this property, the Depth-First traversal strategy, and a decontaminating leader (i.e., an agent), this paper demonstrates that a tree requires at least $\frac{2h}{(t+1)}$ agents, where h is the height of the tree, and t is the immunity time. We can see that the temporal immunity has a significant impact on the process of decontamination.

Hypercube. The hypercube is thoroughly studied in [19]. This paper discusses several strategies for decontaminating a hypercube. The authors consider the local and visibility models, with and without cloning as we mentioned in the previous section.

Mesh. As seen in [23], to decontaminate a mesh (grid) of $M \times N$ nodes, there two possible strategies. The authors assume an asynchronous environment, continuous search, single homebase, and monotone protocol. In the first strategy the authors use the local model in which the agents cannot see neighbouring states but have the ability to exchange information using whiteboard. In this strategy there is a need for a synchronizer that coordinates the protocol execution. Beginning from the homebase, M agents (except one agent that remains at the homebase) move south in order to guard each node in the first column. The synchronizer then coordinates the moves of all agents in such a way that all the agents are moving east at the same time. This process goes on column by column until the whole graph is decontaminated. This protocol requires $M + 1$ agents and $\frac{M^2+4MN-5M-2}{2}$ moves. In the second strategy, the authors use the visibility model, eliminating the need for a synchronizer. In this case, agents depend on what has been written in the whiteboard to decide the next step. This strategy requires M agents (no synchronizer here) and $\frac{M^2+2MN-3M}{2}$ moves.

In the previous two protocols, the authors assume a mesh M×N, where $M \leq N$, so the protocols carry on column by column. In the case of $N < M$, it is better to proceed row by row to obtain a lower number of agents.

Tori. A torus $h \times k$ ($h <= k$) is a mesh where the nodes in the last row are connected to nodes in the first row, and nodes in the first column are connected to nodes in the last column. In order to decontaminate a torus we can use the strategies used to decontaminate a mesh with some slight modifications. In [18], Flocchini et al. use strategies similar to those used for the mesh. Instead of deploying agents to cover one column, agents are deployed to cover two successive columns. One column of agents then stays to guard the nodes from recontamination while the other column of agents moves through the torus. In the local model there is a need for a synchronizer. The complexity of this strategy is $2h + 1$ agents and $(2hk - 4h - 1)$ moves. In the visibility model, there is no need for a synchronizer, and the complexity is $2h$ agents and $(hk - 2h)$ moves. In [29], the authors demonstrate that in order to decontaminate a k-dimensional tori $(k > 2)$ in the presence of local immunity, we need at least $2k$ synchronous agents and $(n + 2k - 2k - 2)$ moves.

Rings. When we have a ring it is easy to find the minimum number of agents required for monotone decontamination.

In the local model, two agents are sufficient to perform the decontamination if they start from a single homebase and move in opposite directions until they are reunited. The number of moves would therefore be n, where n represents the number of nodes, if we assume that an agent needs one time unit to traverse a link. In the visibility model, the two agents move in opposite directions until they reach two consecutive nodes. The number of moves is therefore $n - 1$ (see [29]).

Chordal Rings. In [18], Flocchini et al. provide the results of decontaminating a chordal ring $C_n\{d_1 = 1, d_2, ..., d_k\}$ using the local and visibility models. In the local model, the complexity required to decontaminate the chordal ring using asynchronous execution is $(2d_k + 1)$ agents and $4n - 6d_k - 1$ number of moves. In the visibility model, the number of agents is $(2d_k)$ and the number of moves is $(n - 2d_k)$.

2.3 Black Virus Decontamination

This term was initially introduced by Cai el al. in [5]. The *Black Virus* (BV) is a hostile node that resides in an unknown location, causing the destruction of any visiting mobile agent. The black virus also causes more damage by moving to neighbouring nodes. Therefore, we need a strategy to locate the hostile node and to disinfect the entire topology from its spread. In [5], the authors present the *Black Virus* (BV) threat by combining the statical feature of the *Black Hole* and the mobility feature of the *Intruder Capture*. They also propose a solution to the BVD problem using a team of mobile agents.

The authors assume that there is initially only one BV in the network. Like BH, the location of BV is unknown a priori and the BV stays inactive (unharmful) unless it is triggered. The BV is activated when a mobile agent arrives at its location. The mobile agent is then destroyed and the BV clones itself into as many BVs as its number of degrees. The copies retain the same features as the original and each copy moves to a neighbouring node and remains inactive until triggered. According to [5], the *black virus* is only deactivated when it moves to a node that is occupied by an agent. To the best of our knowledge, this problem has only been studied by Cai et al. in [5].

In [5], the proposed protocol consists of two phases: '*shadowed exploration*' and '*surround and eliminate*'. As the names suggest, the first phase involves exploring the network, locating the *black virus* and triggering it. The second phase involves surrounding the newly created BVs and then triggering them to remove them from the network. This protocol is monotone, meaning that once a node is explored, it is immune from recontamination. The measures of efficiency considered include: the spread (*spread*), the number of agents (*size*) and the number of moves. The BVD problem is solved when the network is completely decontaminated and at least one agent survives. The authors conduct their research using common topologies such as rings, multi-dimensional grids, multi-dimensional tori and hypercubes.

Ring(R). The authors demonstrate that, regardless of the number of nodes (n), the monotone protocol requires ($size(R) = 4$) and ($spread(R) = 2$). It is possible that these results are not quite accurate. If the *black virus* resides in node $n-1$, we either have a non-monotone protocol in which an explored node gets contaminated or less complexity where ($size \leq 4$) and ($spread \leq 2$).

Grid(G). The paper considers a 2-dimensional grid and a q-dimensional grid. In a 2-dimensional grid of size $d_1 \times d_2$, they prove that regardless of n their proposed optimal algorithm would cost ($spread(G) = 3$), ($size(G) = 7$), and at most ($5n + O(1)$) number of moves . While in a q-dimensional grid, the complexity would be ($spread(G) = q + 1$), ($size(G) = 3q + 1$), and at most ($O(qn)$) number of moves.

Torus(T). According to the protocol, decontaminating a q-dimensional torus from BV requires ($spread(T) = 2q$), ($size(T) = 4q$) and ($O(qn)$) moves.

Hypercubes(H). According to the protocol, decontaminating a hypercube from BVs requires ($spread(H) = 3$), ($size(H) = 6$) and at maximum of 34 moves.

Chapter 3

Definitions and Terminology

In this chapter we introduce the primary definitions and terminologies needed for the remainder of this thesis. In particular, we define the mobile agent model, the black virus disinfection problem, and the chordal ring topology. We also describe a general technique used to solve this problem that will serve as the basis for the solutions described in the rest of the thesis.

3.1 The Agent Model

Mobile agents are computational entities that are able to move from one node to a neighbouring node. Some features of mobile agents include processing units, limited memory, the ability to communicate with one another, and the ability to clone themselves. Agents are categorized based on their behaviour according to a specific set of rules. Some of the agents start from an arbitrary node called *homebase* HB, while the rest are created at any node.

The agents we employ are all identical except for the *leader* which has particular capabilities. All of the other agents are given different roles, as we will see later in this chapter.

In this model we assume that the environment is synchronous, meaning that agents move from one node to another in one unit of time and that computation and communication time are insignificant. In the case of an asynchronous network where moving, calculation and communication times are finite but unpredictable, our results still hold with slight modifications, as we will discuss later on.

3.2 The Problem

The *black virus* disinfection problem is a combination of the characteristics of two problems that have been studied extensively in the literature: *Black Hole Search* and *Decontamination.* First, let us briefly recall these problems, already discussed in Chapter 2.

A *Black hole* is a hostile static node, BH, that destroys any visiting agent without leaving a trace. The *Black Hole Search* problem consists of having a team of agents explore an unsafe connected network.

The main issue regarding black hole search is locating the hostile node within reasonable cost. Cost is usually measured in terms of time, number of movements and the size of the team. The static nature of the *black hole* makes it destructive only to visiting mobile agents and not the other nodes in the network. As described in Chapter 2, this problem has been widely studied in different topologies and settings.

Decontamination is the cleaning process after a network has been contaminated by a moving object. This contamination process results in dysfunctional performance. This problem has also been studied under different names: *intruder capture*, *graph search* and *decontamination.* Decontamination can be done using a team of mobile agents that clean a topology in such away that it will not be re-contaminated. As described in Chapter 2, this problem has been studied extensively in different topologies and settings.

The *black virus* disinfection problem is a combination of the static feature of the *Black*

Hole and the mobility feature of the *intruder capture*. We assume that there is initially only one *black virus* in the network. Like BH, the location of BV is unknown a priori and the BV stays inactive (harmless) unless it is triggered. The *black virus* is triggered if a mobile agent arrives at its location. Once it is triggered, the mobile agent is destroyed, the *black virus* clones itself onto other *black viruses* with the same capabilities and each copy moves to a neighbouring node and stays inactive until triggered. If a black virus moves into a node occupied by an agent, it is deactivated and thus removed from the system. In summary, an agent that moves into a node containings a *black virus* is destroyed while a *black virus* that moves into a node occupied by an agent is deactivated. This problem was first introduced by Cai etal. in [5].

The problem for the team of agents involves locating the *black virus* and fully disinfecting the network. Since its position is unknown, it is necessary to trigger it in order to locate it. The first phase of any solution protocol would involve exploring the network, locating the *black virus* and eventually triggering it. Once triggered, the next step involves neutralizing the newly created *black viruses* , whose locations are now known, and restoring the network to a safe state.

One important property of our solution protocol is that it is monotone, meaning that once a node is explored, it stays clean and never gets re-infected.

3.3 The Team of Agents

We now introduce some terminology related to our agents and their role in the protocol. We will be using an *explorer*, *shadow agents*, *surrounding agents* and *cleaning agents.* Their roles will become clearer once we describe the algorithm. In the following section we give an overview of their individual tasks. The team of agents consists of:

- LEA

The Leading Exploration Agent coordinates the operations of our protocol. It starts off from the *homebase*. It has some special capabilities that the other agents do not have including the ability to create new agents, and knowledge of the topology. As we will see, LEA will perform some or all of the following tasks: exploring, routing and terminating.

- CA

  Cleaning Agents are created by the LEA and are deployed to decontaminate the network from the black virus. They serve as the "casualties" of the protocol, performing the actual disinfection of the black virus and disappearing into it. In other words, they are the agents that the LEA sends to the *black viruses* in order to activate them.

- EA

  This is a special cleaning agent that works with LEA to perform what we call *safe exploration* until it comes in contact with the original black virus.

- SH

  Shadow Agents are agents that are created at the homebase and their main role is to protect the nodes that have already been explored from getting infected.

- SA

  Surrounding Agents are created by LEA. They are deployed to surround black viruses triggered by the original one.

In our protocol, we assume that agents' roles sometimes overlap; for instance, a shadow agent becomes a surrounding agent.

3.4 Complexity

The complexity of disinfecting a network C is determined according to three measures: the number of *black viruses* have been created, the total number of agents and the number of moves performed.

- **Number of black viruses**: $Spread(C)$.

  This measure refers to the total number of *black viruses* that have been created including the original one.

- **Number of agents**: $Size(C)$.

  This measure refers to the total number of agents, regardless of their function, and includes the LEA, shadow agents SH, surrounding agents SA and cleaning agents CA.

- **Number of moves**: $Move(C)$

  This measure refers to the total number of moves required by agents to perform their given tasks.

3.5 The Topology

A **Chordal Ring** is a graph with n nodes $v_0, v_1, ..., v_{n-1}$ and link structure $\langle d_1, d_2, ..., d_m \rangle$, $d_i < d_{i+1}$, with $d_m < \lfloor \frac{n}{2} \rfloor$, where each node v_i is adjacent to nodes $v_{(i+d_j)} \mod n$ and $v_{(i-d_j)} \mod n$ for $1 \leq j \leq m$. $N(v_i)$ represents the set of neighbours of node v_i and $|N(v_i)| = 2m$. In the following all indices are assumed to be modulo n and, for simplicity, the modulo will be omitted.

Throughout this thesis the *chordal ring* we refer to has $d_1 = 1$, which means that it is an augmented ring and will be denoted by $C_n(d_1 = 1, d_2, ..., d_m)$. The links of the chordal ring are labeled with chordal sense of direction. For example, the link (v_i, v_j) is labeled with the distance $(j-i) \mod n$ between v_i and v_j along the ring connection. We say that a chordal ring is *shortly chorded* if the largest chord is much smaller than n ($d_m << n$).

We use the (+) sign to indicate a clockwise direction and the (-) sign to indicate a counter-clockwise direction.

Next we will identify particular types of chordal rings that will be studied in the remainder of the thesis:

- A *Double Loop* is a chordal ring $C_n(1,k)$, with $2 < k < \lfloor \frac{n}{2} \rfloor$. In other words, it is an augmented ring where each node has two additional chords at distance of $\pm k$. Therefore, each node has four neighbours: two in the counter-clockwise direction and two in the clockwise direction.

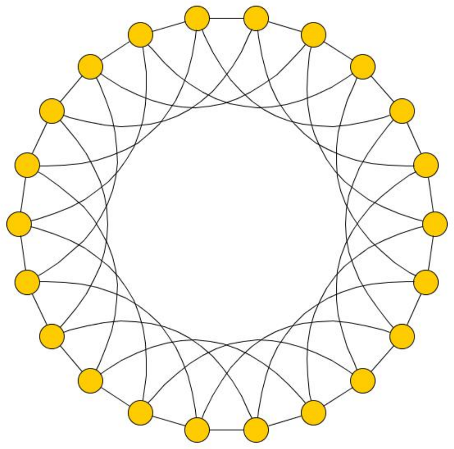

Figure 3.1: A double Loop chordal ring C(1,4).

- A *Triple Loop* is a chordal ring $C_n(1,p,k)$, with $p < k$ and $k < \lfloor \frac{n}{2} \rfloor$. In other words, it is an augmented ring where each node has two additional chords at a distance of $\pm p$ and $\pm k$. Therefore, each node has six neighbours: three in the counter-clockwise direction and three in the clockwise direction.

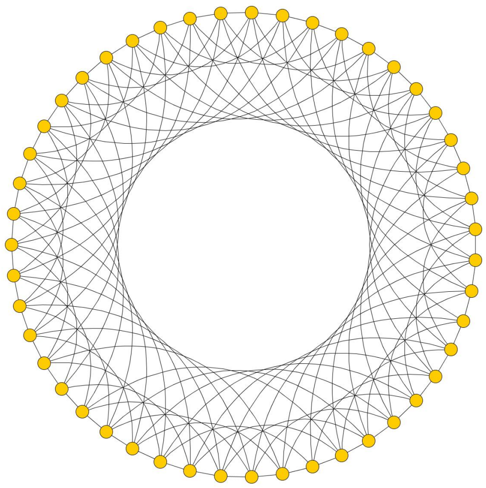

Figure 3.2: A triple Loop chordal ring C(1,5,9).

- A *Consecutive-Chords* is a chordal ring $C_n(1, 2, \ldots, k-1, k)$, with $k < \lfloor \frac{n}{2} \rfloor$. In a chorded ring, each node has $2k$ consecutive neighbours: k neighbours in each direction.

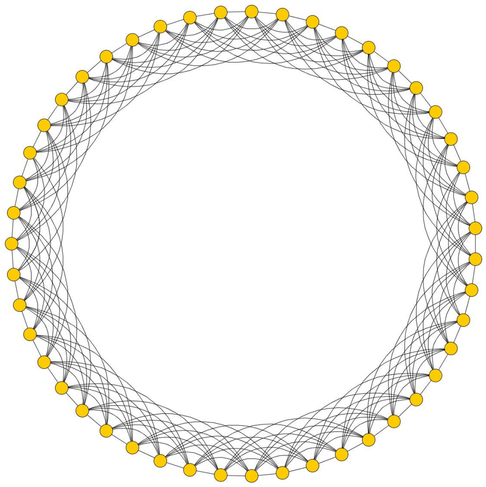

Figure 3.3: A consecutive-chords chordal ring C(1,2,3,4,5).

3.6 General Strategy

We now describe a general high level strategy that can be employed to solve the *black virus* decontamination problem. Our algorithm can be divided into two phases: *Exploring and*

Shadowing and *Surrounding and Eliminating.*

3.6.1 Phase 1: Exploring and Shadowing

The main goal of this phase is to determine the location of the original *black virus*. Since the location of the *black virus* is unknown a priori, we need to explore the topology until it is found. Finding the node in which the *black virus* resides implies triggering that *black virus*, creating new ones, destroying an agent in the process, and cleaning the node that contained the original *black virus*. This process is done by the leading exploration agent LEA, the exploring agent EA and a group of shadow agents SH.

Starting from the *homebase* , the leader and the exploring agent explore the chordal ring by moving throughout the outer ring only. The assumption here is that the leader and the exploring agent are moving in a clockwise direction using the *safe exploration* technique: LEA and EA are at safe node v_j, EA moves to the next node v_{j+1} while LEA waits at v_j; if EA returns to its leader then v_{j+1} is not a *black virus* and they both move to v_{j+1}. If, while moving in this fashion, LEA receives a *black virus* instead of EA, then the location of the original *black virus* is detected and EA is destroyed.

If node v_i represents the node under exploration, $N(v_i)$ represents the neighbours of v_i, and $N_{ex}(v_i)$ and $N_{un}(v_i)$ denote the set of explored and unexplored neighbours of node v_i. The role of the shadow agents is to protect the already explored nodes from potential reinfection. When a new node v_i is under exploration, the shadow agents should occupy its explored neighbours $N_{ex}(v_i)$. Once EA arrives to the *black virus* location it is destroyed, LEA receives a BV instead of EA, that node is cleared, and the new *black viruses* are now located on all the unprotected (unexplored) neighbours of the *black virus* . The set of nodes between the *homebase* and the node being explored is called the *safe area* and denoted by S_{area}. For our protocol to be monotone, we have to insure that the nodes in the safe area do not get re-infected. To do so, the *shadow agents* are deployed to guard the explored counter-clockwise neighbours of the next node to be visited. The deployment of SHs begins when LEA and EA have explored at least d_2 nodes. In other words, if $C_n(d_1 = 1, d_2, ..., d_k)$ and $|S_{area}| \geq d_2$, at least one SH is deployed and the number of SHs increases to match the number of neighbours of the currently explored node in the safe area. There is a section of the *safe area* called the *danger area*, denoted by D_{area}, where $v_{n-1} \geq D_{area} \geq v_{n-d_k}$. In this area, the possibility of reinfection of explored nodes is increased. Because of this, we need SHs to guard the counter-clockwise neighbours as well as some or all of the clockwise neighbours.

Since all operations are synchronized, shadow agents are created at once and stay inactive until their timer reaches zero. In other words, all SHs have timers that have been set to different values in order to synchronize them with the exploring team (LEA and EA). When the *black virus* is found the SHs are notified by the LEA. Figure 3.4 shows an example of the first phase in a double loop. The green nodes represent the safe area and the red nodes represent the danger area.

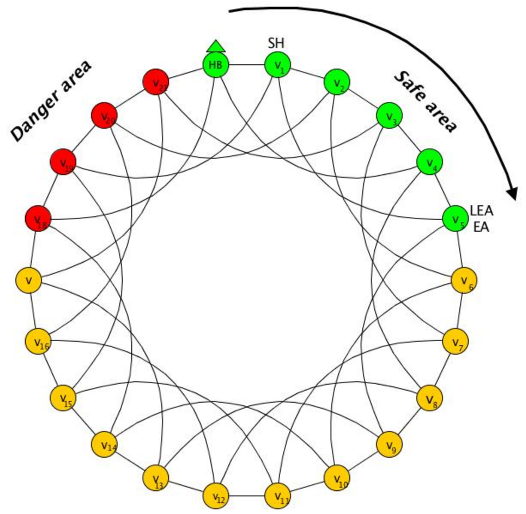

Figure 3.4: First phase in a double loop chordal ring C(1,4).

```
EXPLORING AND SHADOWING

let $HB = v_0$
Agents $EA$ and LEA at safe node $v_i$.
- Compute $N_{ex}(v_{i+1})$
- For each $q \in N_{ex}(v_{i+1})$
    $SH$ is deployed
- $EA$ moves to $v_{i+1}$
    If $EA$ returns back to $v_i$
        $LEA$ and $EA$ move to $v_{i+1}$
    Else (i.e., $BV$ moves to $v_i$)
        $EA$ is destroyed
        $BV = N_{un}(v_{i+1})$
```

This phase comes to an end when the *black virus* is detected and cleared; meanwhile, other *black viruses* have been created and moved to the unexplored neighbours of the original *black virus.* At this point, the second phase begins. The first phase is common to all of the strategies proposed in this thesis.

3.6.2 Phase 2: Surrounding and Eliminating

Once the *black virus* node is detected, LEA moves to its location and the *Surrounding and Eliminating* phase begins. In this phase, the entire chordal ring is disinfected. The disinfection process is conducted by sending agents to the unexplored neighbours of the newly created *black viruses.* LEA then sends CA(s) to activate those *black viruses* and clear them.

First, let us introduce some notation. x_0 represents the original *black virus* . V represents the set of all vertices. The set of black viruses in the system is denoted by $\mathcal{BV}$.

$\mathcal{S} = V - \mathcal{BV}$ represents the set of nodes that does not contain any $\mathcal{BV}$. $\mathcal{T}$ represents the set of targets, that is, the nodes to be occupied.

```
SURROUNDING AND ELIMINATING

LEA and $SH$s covering all $N_{ex}(v)$
$BV$ comes back from $v$.
    - Compute $N_{un}(v)$
    - For each $u \in N_{un}(v)$:
        Deploy an agent to each $z \in \{N(u) \setminus N_{un}(v)\}$
        When $N(u)$ is covered:
            Deploy one agent to $u$
```

In this phase, we need the same number of SAs as the number of neighbouring nodes of the *black viruses*. The SAs move to their destinations, and once all of the agents are in position, LEA sends CAs to trigger all the *black viruses* at once.

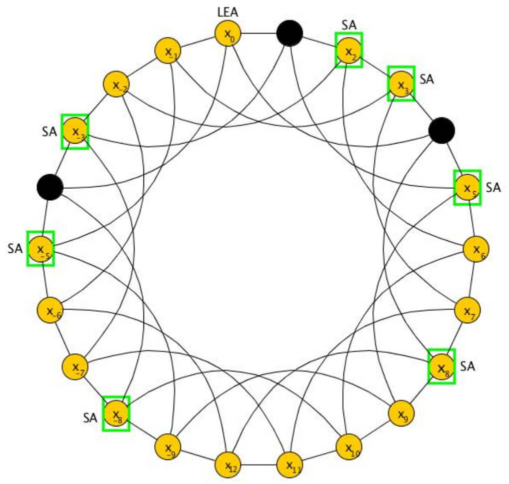

Figure 3.5: Location of SAs in $C(1, 4)$ where the black nodes represent the triggered black viruses.

Figure 3.5 shows the location of SAs in a double loop after triggering the original *black virus* which then creates three more *black viruses* in the worst case.

Deploying agents is the most important part of this phase. Deployment can be done in different settings. In this thesis, we discuss two deployment strategy variations: *local* and *non-local*.

The non-local strategy that we propose is called *Move-Optimal Deployment*. In this strategy, we assume that LEA is the manager of the entire process. LEA resides in node x_0, the location of the original *black virus*, where it sends the surrounding agents to their destinations through the shortest path. Therefore, a surrounding agent carries the whole path starting from x_0 to any $t \in \mathcal{T}$. It is possible to devise a variant of this optimal strategy in which all agents have full topology information. In this case, there is no need for a surrounding agent to carry the full path while moving toward its target since the agent can re-compute it at each intermediate node.

The local strategies are based on the *Greedy* approach. Some of the greedy strategies are only applicable to certain chord structures because they would not produce the correct routing when used with other structures. These strategies do not require as much storage as the non-local strategy since the surrounding agents do not carry paths to their destinations. Instead, they calculate their next step each time on the basis of their current location and the location of the target destination with no additional information. The local strategies that will be discussed in this thesis are: the *Simple Greedy*, the *Smart Greedy* and the *One-direction Greedy*. In the *Simple Greedy* strategy, at each node the agent must decide the next node according to its distance from the target. In the *Smart Greedy* strategy, the agents use more information in order to obtain better results. In the *One-direction Greedy* strategy, the agents move greedily in one direction until they reach their target. We will discuss each strategy in depth throughout this thesis.

3.7 Conclusions

In this chapter we introduced our problem and some important definitions and terminologies. We also described the team of agents and their specific roles: coordinating, exploring, shadowing, surrounding and activating. Moreover, we discussed the main topology of this thesis which is the chordal ring. Finally, we presented a general overview of the two phase solution: *Exploring and Shadowing* and *Surrounding and Eliminating*.

Chapter 4

Black Virus Disinfection in Double Loops

In this chapter we consider the process of using agents to locate and clear the *black virus* in the *double loop* chordal ring $C_n(1, k)$ in a *synchronous* environment. The *Double Loop* ring is an interconnected ring of n nodes $v_0, \ldots v_{n-1}$, where each node is connected to two additional neighbours at distance of k. The neighbourhood of node v_i is thus given by: $N(v_i) = \{v_{i+1}, v_{i+k}, v_{i-1}, v_{i-k}\}$.

As with all the other topologies, the leader and the exploring agent locate the *BV* and then surrounding agents are sent to surround the new triggered viruses and eliminate them so that the entire topology is disinfected. We propose several variants of the main strategy depending on the routing procedure used to surround the new triggered viruses. We evaluate the complexity of our strategies by considering the overall number of agents employed, the number of agent casualties and the number of moves required. The table below summarizes the worst case complexities of the three main strategies.

Double Loop	Spread W.C	Agents W.C	Moves
Move-Optimal	4	12	$3k + 31$
Simple Greedy	4	12	$6k + 12$
Smart Greedy	4	12	$5k + 17$

4.1 Exploring and Shadowing

This phase is common to all of the strategies. The two agents, LEA and EA, explore each node on the outer ring in a clockwise direction starting from the homebase using the *Safe Exploration* technique: LEA and EA are at node v_j. EA moves to the next node v_{j+1} while LEA waits at v_j. If EA returns to its leader they both move to v_{j+1}, otherwise, the BV is detected and EA is destroyed. In order to diminish the effect of the BV, a shadow agent SH is deployed to guard the explored neighbours of the next node to be visited. This cannot happen unless LEA and EA have passed through at least k nodes. In other words, if the safe area is $\geq k$.

Lemma 1. *The Exploring and Shadowing phase is a monotone protocol that locates the black virus correctly.*

Proof. The exploring team follows the outer ring and EA always precedes LEA, so that the *black virus* is eventually detected. The monotonicity of this phase is apparent in the presence of shadow agents that were all created in the beginning at the homebase and then synchronize their movements with the exploring team. $\square$

```
EXPLORING AND SHADOWING

let $HB = v_0$
Agents $EA$ and LEA at safe node $v_j$.

if $(k \leq j+1 < n-k)$
    $N_{ex}(v_{j+1}) = \{v_j, v_{j+1-k}\}$
    1 $SH$ is deployed to protect $v_{j+1-k}$
if $(n-k \leq j+1 < n-1)$
    $N_{ex}(v_{j+1}) = \{v_j, v_{j+1-k}, v_{j+1+k}\}$
    2 $SH$s are deployed to protect $v_{j+1-k}$,$v_{j+1+k}$
if $(j+1 = n-1)$
    $N_{ex}(v_{j+1}) = \{v_j, v_{j+1-k}, v_{j+1+k}, v_0\}$
    3 $SH$s are deployed to protect $v_{j+1-k}$,$v_{j+1+k}$,$v_0$

EA moves to $v_{j+1}$.
```

Some observations can be made following the implementation of this strategy:

Theorem 2. *In the worst case scenario, the black virus is detected in $4n - 7$ moves.*

Proof. The worst case for the number of moves is when the *black virus* is found at node (v_{n-1}) after exploring $n-1$ nodes. The BV triggers no new *black viruses* since all the neighbours are already explored in the safe area and all are protected by SHs.

The complexity of this case is $3(n-1)-2$ for the movement of LEA and EA, $n-1-k$ for one SH and $(k-1)$ for the other SH. □

Theorem 3. *In any double loop chordal ring C, the worst case scenario in term of the number of agents required occurs when three new black viruses are created after triggering the original virus.*

Proof. The worst case for the number of *black viruses* created upon activation of the original is when that one is found at node v_i where $1 \leq i < k$. In this case, the *black virus* (x_0) triggers three new *black viruses*: x_1, x_k and x_{-k} because no SH has been deployed at this time. Note that x_{-1} is always occupied by LEA. □

4.2 Surrounding and Eliminating

In the double loop chordal ring, when the BV is triggered it may affect up to three neighbouring nodes. If x_0 represents the original BV, node x_{-1} is clearly protected by LEA. x_1, x_k and x_{-k} are protected only if they belong to the safe area and are occupied by shadow agents SH. In summary, the best case scenario in terms of the number of *black viruses* occurs if no more BVs are created after EA triggers the original black virus. The worst case scenario involves the creation of three new BVs.
To handle the spread of the BVs, LEA has to send agents to surround and clear those faults. The only way to destroy BVs is when they arrive at guarded nodes.

To do that, LEA creates agents that are sent to specific targets, the neighbours of the new created BVS. Once all the agents are in position, LEA sends cleaning agents (casualties) to trigger all the BVs at once. This means that we require the same number of SAs as the number of neighbours of BVs and as many CAs as BVs.

Lets take a look at the necessary notation. x_0 represents the discovered BV. V represents the set of all vertices. The set of black viruses in the system is denoted by $\mathcal{BV}$. $\mathcal{S} = V - \mathcal{BV}$ represents the set of clean nodes (not containing any BV). $\mathcal{T}$ represents the set of targets (the nodes to be occupied). S_{area} represents the *safe area* and $|S_{area}|$ is the number of nodes in that area. D_{area} represents the *danger area* where recontamination needs to be avoided.

```
SURROUNDING AND ELIMINATING

LEA and $SH$s covering all $N_{ex}(v)$
$BV$ comes back from $v = x_0$.


if ($|S_{area}| < k$) (* LEA is covering $x_{-1}$ *)
    $N_{un}(x_0) = \{x_1, x_k, x_{-k}\}$
if ($k \leq |S_{area}| < n - k$) (* LEA and $SH$ covering $x_{-1}, x_{-k}$ *)
    $N_{un}(x_0) = \{x_1, x_k\}$
if ($n - k \leq |S_{area}| < n - 1$) (* LEA and $SH$s covering $x_{-1}, x_{-k}, x_k$ *)
    $N_{un}(x_0) = \{x_1\}$
Else (* LEA and $SH$s covering $x_{-1}, x_{-k}, x_k, x_1$ *)
    $N_{un}(x_0) = \emptyset$


All $SH$ make one move in clockwise direction.
For each $u \in N_{un}(x_0)$:
    DEPLOY an agent to each $z \in \{N(u) \setminus N_{un}(x_0)\}$
When $N(u)$ is covered:
    DEPLOY one agent to $u$
```

From the previous algorithm we can see that once a SH receives a *black virus* it moves to the neighbouring node through chord +1 in order to be part of the surrounding team. Thus, we can see that agents sometimes change roles.

Synchronicity is one of the assumptions of this model. All agents are synchronized and it takes one time unit to traverse a link connecting two neighbouring nodes.

It is convenient for our purposes to divide the chordal ring into three segments starting from the homebase as seen in figure 4.1.

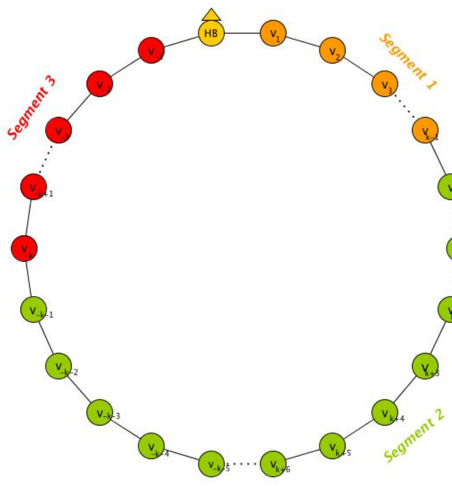

Figure 4.1: Dividing the double loop into three segments.

- *Segment 1* contains nodes $v_1, \dots v_{k-1}$. If the *black virus* is in this segment, the size of the safe area is $1 \leq |S_{area}| \leq k-1$.
- *Segment 2* contains nodes $v_k, \dots v_{n-k-1}$. If the *black virus* is in this segment, the size of the safe area is $k \leq |S_{area}| \leq n-k-1$.
- *Segment 3* contains nodes $v_{n-k}, \dots v_{n-1}$. If the *black virus* is in this segment, the size of the safe area is $n-k \leq |S_{area}| \leq n-1$.

We will see that complexity changes depending on the location of the black virus relative to the segment.

The number of agents required to disinfect the double loop chordal ring C is the same regardless of the strategy employed, whereas the number of moves varies depending on the deployment method.

Theorem 4. *Regardless of deployment strategy and chord length, a maximum of 12 agents are employed in any double loop $C_n(1, k)$ for the black virus disinfection protocol.*

Proof. The number of agents required is determined by the location of the original black virus, regardless of the deployment strategy.

- When the BV is located at any node in *Segment 1*, we would obtain the worst complexity in terms of CA and SA since activating the *black virus* will create three more *black viruses* at x_1, x_k, and x_{-k}. LEA then moves to x_0 and deploys seven SAs to occupy $x_2, x_{k-1}, x_{k+1}, x_{2k}, x_{-k+1}, x_{-k-1}$ and x_{2k}. The total number of agents employed is then 12: 4 CAs, 7 SAs and one LEA. Thus, in this case, $Spread(C) = 4$ and $Size(C) = 12$.

- If BV is found in *Segment 2* , activating the original *black virus* would create two more *black viruses* at x_1 and x_k because x_{-k} is guarded by a SH. SH then moves to x_{-k+1} and LEA moves to x_0. LEA then deploys 4 SAs to occupy x_2, x_{k-1}, x_{k+1} and x_{2k}. The total number of agents employed is then 9: 3 CAs, 4 SAs, 1 SH and LEA. Thus, in this case, $Spread(C) = 3$ and $Size(C) = 9$.

- If BV is found in *Segment 3*, there are two possible scenarios:

  – When $n - k \leq i < n - 1$:

    This segment is part of what is called the *Danger area* (D_{area}). If the BV is located at any node in this segment, activating the original *black virus* would create only one *black virus* at x_1, while x_{-k} and x_k are guarded by SHs. Subsequently, in addition to the SH at x_{-k+1}, the SH at x_{k+1} and LEA at x_0, the LEA deploys 1 SA to occupy x_2. The total number of agents employed is then 6: 2 CAs, 1 SA, 2 SH and one LEA. Thus, in this case, $Spread(C) = 2$ and $Size(C) = 6$.

  – When $i = n - 1$, no more *black viruses* are created since all neighbouring nodes of the *black virus* are guarded by SHs and LEA. There are then no SAs to be deployed and all the moves are done in the first phase. The total number of agents employed is then 5: 1 CA, 3 SHs and one LEA. Thus, in this case, $Spread(C) = .1$ and $Size(C) = 5$.

□

For the deployment phase we propose two types of strategies: non-local and local. In non-local strategies, LEA calculates the shortest path to reach the various targets and gives each agent the information about the corresponding path. In local strategies, each SA decides the next node locally, based on the indication of the destination target.

4.2.1 Move-Optimal Deployment

In this section we describe a non-local approach where SAs follow paths set up by their leader LEA who has full topological knowledge. Once the *black virus* is triggered, LEA, knowing the topology and the location of the black viruses, computes the best route to each target and gives the information to the corresponding SAs. Because of the simple structure of the chordal ring, we can calculate the exact length of the shortest paths to reach the target and verify experimentally that they are indeed optimal.

In this section we describe the path to be followed by each agent for the simpler case of shortly-chorded double loops, and then for the case of general double loops. Before we describe these two situations we will identify some special constant size paths that can be used to reach all targets in both situations. The special path to reach node x_i from x_0 is denoted by σ_i. Depending on the chord structure, it may be possible to devise shorter paths in some cases. A detailed analysis of all possible situations is carried out in the next two sections.

Target x_i	Special Path σ_i
x_{k-1}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k} x_{k-1}$
x_2	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k} x_{k-1} \xrightarrow{+k} x_{2k-1} \xrightarrow{+1} x_{2k} \xrightarrow{+1} x_{2k+1} \xrightarrow{+1} x_{2k+2} \xrightarrow{-k} x_{k+2} \xrightarrow{-k} x_2$
x_{2k}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k} x_{k-1} \xrightarrow{+k} x_{2k-1} \xrightarrow{+1} x_{2k}$
x_{k+1}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k} x_{k-1} \xrightarrow{+k} x_{2k-1} \xrightarrow{+1} x_{2k} \xrightarrow{+1} x_{2k+1} \xrightarrow{-k} x_{k+1}$
x_{-k-1}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-k} x_{-k-1}$
x_{-k+1}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-k} x_{-k-1} \xrightarrow{-k} x_{-2k-1} \xrightarrow{+1} x_{-2k} \xrightarrow{+1} x_{-2k+1} \xrightarrow{+k} x_{-k+1}$
x_{-2k}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-k} x_{-k-1} \xrightarrow{-k} x_{-2k-1} \xrightarrow{+1} x_{-2k}$

4.2.1.1 Shortly-chorded Double Loops

A shortly-chorded double loop is a double loop where the size is significantly bigger than the length of the largest chord, that is: $k << n$. All results of this section hold for $n > 4k$.

After triggering the original BV (x_0) the neighbouring nodes are in one of two states: guarded or contaminated. Node x_{-1} is guarded by LEA. Depending on the size of the *safe area*, nodes x_1, x_k and x_{-k} could be in either state. As mentioned earlier, the *black virus* could be located in one of three segments.

For some of the targets it is easy to identify the shortest path from x_0. For example, x_{k-1} cannot be reached directly but it can be reached in two moves from x_0 through node x_{-1} (i.e., following the special path σ_{k-1}). For other targets it is harder to determine whether σ_i represents the best alternative.

We now consider the different routes to each target in four different cases depending on the location of the *black virus* where $\pi[x_0, x_i]$ denotes a path to reach target x_i from x_0.

- **Case 1**: Consider the case where the *black virus* is in the second segment of the chordal ring, that is, when $k \leq |S_{area}| < n - k$. In this case, triggering the original *black virus* creates two more *black viruses*: x_1 and x_k, and thus $\mathcal{T}$=$\{x_2, x_{k-1}, x_{k+1}, x_{2k}\}$.

- x_{k-1}: Node x_{k-1} is reached through σ_{k-1}, which is clearly the shortest path.
- x_2: Since for $k = 3$ we know that $x_2 = x_{k-1}$, we consider $k > 3$.

  Depending on the length of k, x_2 could be reached optimally in different ways (note that all of them are actually shorter than the special path σ_2 identified earlier):

  * If $k = 4$ and $x_{k-1} = x_3$, the shortest path is the following:

  $$\pi[x_0, x_2] = \pi_1 = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k} x_{k-1} = x_3 \xrightarrow{-1} x_2$$

  * If $k > 4$, taking advantage of the fact that x_{-k} is known to be safe:

  $$\pi[x_0, x_2] = \pi_2 = x_0 \xrightarrow{-k} x_{-k} \xrightarrow{+1} x_{1-k} \xrightarrow{+1} x_{2-k} \xrightarrow{+k} x_2$$

- x_{2k}: Node x_{2k} is reached through σ_{2k}.
- x_{k+1}: If $k > 4$, x_{k+1} is reached through σ_{k+1}. Otherwise, it is reached faster through:

  $$\pi[x_0, x_{k+1}] = \pi_3 = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k} x_{k-1} = x_3 \xrightarrow{-1} x_2 \xrightarrow{+k} x_{k+2} \xrightarrow{-1} x_{k+1}$$

- **Case 2**: When the *black virus* is in the third segment excluding the last node before the homebase (i.e., $n-k \leq |S_{area}| < n-1$), only one *black virus* is generated (x_1) since the rest of the neighbours have been explored and guarded. Thus, $\mathcal{T}$=$\{x_2, x_{k+1}\}$.
  - x_2: If $k \leq 4$, x_2 is reached using π_1, otherwise it is reached using π_2
  - x_{k+1} is reached in one move from node x_k because x_k contains a SH agent, which received a copy of the original *black virus*.

- **Case 3**: We have a special case in which the *black virus* is located on the last node before the homebase. In this case, all neighbours are guarded and no more *black viruses* are created. The surrounding phase is no longer needed.

- **Case 4**: When the *black virus* is in the first segment (i.e., $|S_{area}| < k$) we have to consider the nodes that might be contaminated if they do not belong to the area already protected by shadows. In fact, when $|S_{area}| < k$, LEA has to deploy agents to the neighbours of the three new black viruses $\mathcal{BV}$=$\{x_1, x_k, x_{-k}\}$ while x_{-1} is being guarded by LEA and no SH has been deployed.

  Node x_{-k} has three unguarded neighbours, $\mathrm{N}_{un}(x_{-k})$= $\{x_{-k-1}$,x_{-k+1},$x_{-2k}\}$. Thus, $\mathcal{T} = \{x_2, x_{k-1}, x_{k+1}, x_{2k}, x_{-k-1}, x_{-k+1}, x_{-2k}, \}$.

  We have:

  - Nodes x_{k-1} and x_{2k} are reached through paths σ_{k-1} and σ_{2k}.
  - x_{k+1} is reached through π_3 or σ_{k+1} as explained in **Case 1**.
  - x_{-k-1} is reached through path σ_{-k-1}.
  - x_{-2k} is reached through path σ_{-2k}.
  - x_{-k+1} is reached through path σ_{-k-1}.
  - x_2: In this case, node x_{-k} is a *black virus* and any path that passes through it to reach x_2 should be avoided.
    - ∗ if $k < 8$, the path to reach x_2 is:

    $$\pi[x_0, x_2] = \pi_4 = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k} x_{k-1} \xrightarrow{-1} x_{k-2} \xrightarrow{-1}, ... \xrightarrow{-1} x_2$$

    - ∗ otherwise, (i.e. $k \geq 8$), x_2 is reached through σ_2.

The following theorems summarize the number of moves:

Theorem 5. *In any shortly-chorded double loop $C_n(1,k)$, if $|S_{area}| \geq k$, the number of moves to complete the* Surrounding and Eliminating *phase is a maximum of* 20.

Proof. The first move is made by LEA to move to x_0. Then, by construction, we know that:

- $k \leq |S_{area}| < n-k$
  - Node x_{k-1} is reached in two moves through path σ_{k-1}
  - In the case of node x_2 we have three possibilities: if $k > 4$, the target is reached in four moves; if $k = 4$, the target is reached in three moves; if $k = 3$, x_2 coincides with x_{k-1} and the target is reached in two moves. The maximum amount of moves is thus four.
  - x_{2k} is reached in four moves through path σ_{2k}
  - x_{k+1} follows the same route as x_{2k} in the clockwise direction with the addition of two moves, or from x_2 with the addition of two more moves. The algorithm selects the minimum. The maximum number of moves in this case is then six.
- $n-k \leq |S_{area}| < n-1$

  In this case, one *black virus* is generated (x_1) since the neighbouring nodes are guarded. Thus, $\mathcal{T}$=$\{x_2, x_{k+1}\}$.
  - x_2 is reached in maximum four moves, corresponding to path π_2
  - x_{k+1} is reached in one move since the SH at node x_k received a copy of the original *black virus*, and SHs always make one move in the clockwise direction when they receive a *black virus* in order to be part of the surrounding team.

Since all of the agents are sent at the same time they will arrive at their destinations within six time units. After waiting six time units, the LEA will send two agents to clear the two *black viruses* , thus performing two more moves. □

Theorem 6. *In any shortly-chorded double loop $C_n(1,k)$, if $|S_{area}| < k$, the number of moves to complete the* Surrounding and Eliminating *phase is a maximum of* 36*.*

Proof. One move is made by LEA to move to x_0. Then, by construction, if $|S_{area}| < k$, we know that: node x_{k+1} is reached within six moves through σ_{k+1} or π_3; node x_{k-1} is reached within two moves through σ_{k-1}; node x_{2k} is reached within four moves through σ_{2k}; node x_{-k-1} is reached within two moves through σ_{-k-1}; node x_{-2k} is reached within four moves through σ_{-2k}; x_{-k+1} is reached through x_{-2k} with the addition of two more moves through path σ_{-k+1}, which means a maximum of six moves; node x_2. In this case x_2 is reached through σ_2 or π_4 and the number of moves would be a maximum of eight. After all the agents arrive at their destinations, three more moves are needed to clear the black viruses.

□

4.2.1.2 General Double Loop

In this section, we handle double loop chordal rings in general $C_n(1,k)$, where $2 < k < \lfloor \frac{n}{2} \rfloor$. Finding the exact optimal number of moves in this case is much more complicated and difficulties arise because, depending on the length of the chords and the number of nodes, it may be more convenient to wrap around the ring in order to reach certain targets.

In order to describe the *Surrounding and Eliminating* process we use the same notations and cases. In other words, we have two cases: when $|S_{area}| \geq k$ and when $|S_{area}| < k$. Therefore, the set of targets $\mathcal{T}$ is $= \{x_2, x_{k-1}, x_{k+1}, x_{2k}, x_{-2k}, x_{-k+1}, x_{-k-1}\}$, $= \{x_2, x_{k-1}, x_{k+1}, x_{2k}\}$, $= \{x_2, x_{k+1}\}$ or $\emptyset$. The only difference between this section and the previous one is in the *deployment* part of the algorithm since, as mentioned above, we have more possibilities in the general structure of the double loop.

Case $|S_{area}| \geq k$. Let us first consider the case in which $|S_{area}| \geq k$. For some targets, the path to be followed is the same as the one employed for shortly chorded rings. This is the case for x_{k-1} and x_2. Node x_{k-1} is reached through σ_{k-1}. If $k = 4$, node x_2 is reached through π_1; otherwise, it is reached through π_2.

In the following discussion we will refer to the special paths σ_i identified in the previous section and compare them to other possible routes in order to find the best one.

In order to find the best routes to reach nodes x_{k+1} and x_{2k}, we divide the ring into "windows" of size $(k+1)$ starting from x_0 in both directions, and consider different cases depending on the number of windows covering the ring.

1. $\lceil \frac{n}{k} \rceil > 4$.

   Intuitively, n is big enough for us to consider this a Shortly-chorded double loop and we follow the same paths indicated in the previous section to reach the target.

2. $\lceil \frac{n}{k} \rceil = 4$.

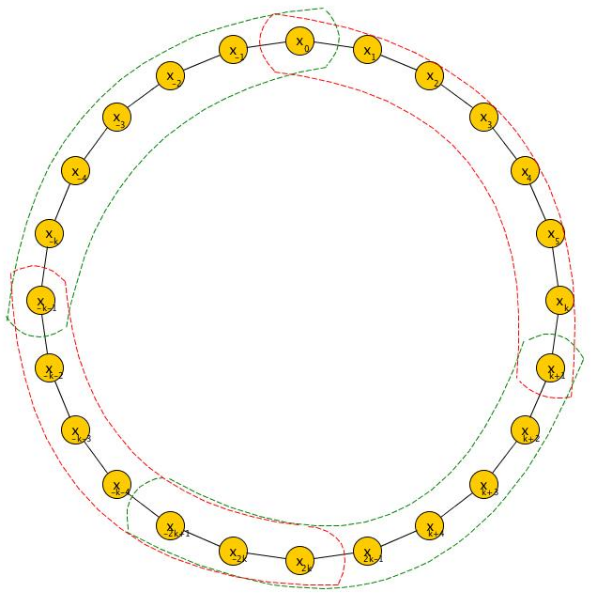

Figure 4.2: Dividing the outer ring into 4 windows of size $k + 1$.

   In this case it might be more efficient to "wrap-around" the chordal ring to reach both x_{k+1} and x_{2k}.

- x_{2k}.

  To reach target x_{2k} we distinguish between different situations depending on the relationship between n and k:

  — If $n = 4k$

  $$\pi[x_0, x_{2k}] = x_0 \xrightarrow{-k} x_{-k} \xrightarrow{-k} x_{-2k} \xrightarrow{-1} x_{2k}$$

  — If $k < \frac{2}{7}n - 1$

  $$\pi[x_0, x_{2k}] = \min\{\pi_5, \sigma_{2k}\}$$

  where

  $$\pi_5 = x_0 \xrightarrow{-k} x_{-k} \xrightarrow{-k} x_{-2k} \xrightarrow{+1} x_{-2k+1} \xrightarrow{+1} ... \xrightarrow{+1} x_{2k}$$

  and σ_{2k} is the special path we referred to in shortly-chorded loops.

  $$\sigma_{2k} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k} x_{k-1} \xrightarrow{+k} x_{2k-1} \xrightarrow{+1} x_{2k}$$

  — If $k \leq \frac{2}{7}n - 1$

  $$\pi[x_0, x_{2k}] = \min\{\pi_6, \sigma_{2k}\}$$

  where

  $$\pi_6 = x_0 \xrightarrow{-k} x_{-k} \xrightarrow{-1} x_{-k-1} \xrightarrow{-1} x_{-k-2} \xrightarrow{-1} ... \xrightarrow{-1} x_{2k}$$

- x_{k+1}

  — If $n = 4k$

  $$\pi[x_0, x_{k+1}] = x_0 \xrightarrow{-k} x_{-k} \xrightarrow{-k} x_{-2k} \xrightarrow{-k} x_{k+1}$$

—Else

$$\pi[x_0, x_{k+1}] = \min\{\pi_7, \sigma_{k+1}\}$$

where

$$\pi_7 = x_0 \xrightarrow{-k} x_{-k} \xrightarrow{-k} x_{-2k} \xrightarrow{-1} x_{-2k-1} \xrightarrow{-1} ... \xrightarrow{-1} x_{k+1}$$

and σ_{k+1} is the path to reach x_{k+1} in shortly-chorded loops.

$$\sigma_{k+1} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k} x_{k-1} \xrightarrow{+k} x_{2k-1} \xrightarrow{+1} x_{2k} \xrightarrow{+1} x_{2k+1} \xrightarrow{-k} x_{k+1}$$

3. $\lceil \frac{n}{k} \rceil < 4$

- x_{k+1}. In this case, in addition to σ_{k+1}, there are three other possibilities and we have:

$$\pi[x_0, x_{k+1}] = \min\{\sigma_{k+1}, \pi_8, \pi_9, \pi_{10}\}$$

  – (* when the $+k^{th}$ neighbour of x_{k+1} is between x_{-1} and x_{-k}, it might be advantageous to move to x_{-1} first *)

$$\pi_8 = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-1} x_{-2} \xrightarrow{-1} ... \xrightarrow{-1} x_{2k+1} \xrightarrow{-k} x_{k+1}$$

  – (* if instead x_{-k} is closer to x_{2k+1} , then it might be better to move to first x_{-k} *)

$$\pi_9 = x_0 \xrightarrow{-k} x_{-k} \xrightarrow{+1} x_{-k+1} \xrightarrow{+1} ... \xrightarrow{+1} x_{2k+1} \xrightarrow{-k} x_{k+1}$$

  – (* finally, if x_{-k} is closer to x_{k+1} *)

$$\pi_{10} = x_0 \xrightarrow{-k} x_{-k} \xrightarrow{-1} x_{-k-1} \xrightarrow{-1} ... \xrightarrow{-1} x_{k+1}$$

- x_{2k}. Target x_{2k} resides between x_{-1} and x_{-k}, so we first have to determine which of them is closer. We have:

$$\pi[x_{11}, x_{k+1}] = \min\{\sigma_{2k}, \pi_{11}, \pi_{12}\}$$

where

$$\pi_{11} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-1} x_{-2} \xrightarrow{-1} ... \xrightarrow{-1} x_{2k}$$

$$\pi_{12} = x_0 \xrightarrow{-k} x_{-k} \xrightarrow{+1} x_{-k+1} \xrightarrow{+1} ... \xrightarrow{+1} x_{2k}$$

Case $|S_{area}| < k$. Consider now the case in which $|S_{area}| < k$. In this case: $\mathcal{T}= \{x_2$, x_{k-1}, x_{k+1}, x_{2k}, x_{-2k}, x_{-k+1}, $x_{-k-1}\}$. Notice here that in all cases, any route that passes through x_{-k} should be avoided because x_{-k} is now a *black virus*.

- Nodes x_{k-1}, x_{-2k}, x_{-k+1}, and x_{-k-1} are reached through the paths σ_{k-1}, σ_{-2k}, σ_{-k+1} and σ_{-k-1} respectively.
- To find the best routes to reach nodes x_2, x_{k+1} and x_{2k}, we also use the concept of "windows" mentioned earlier.
    1. $\lceil \frac{n}{k} \rceil > 4$

       This case is the same as the shortly-chorded chordal ring structure and uses the methodology described in the previous section.
    2. $\lceil \frac{n}{k} \rceil = 4$

       – x_{2k}. (* The only option besides the route σ_{2k} is π_{13} described below, which should be taken only if $n = 4k + 1$. *)

$$\pi_{13} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-k} x_{-k-1} \xrightarrow{-k} x_{2k}$$

- x_{k+1}

$$\pi[x_0, x_{k+1}] = \min\{\sigma_{k+1}, \pi_{14}\}$$

where

$$\pi_{14} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-k} x_{-k-1} \xrightarrow{-k} x_{-2k-1} \xrightarrow{-1} x_{-2k-2} \xrightarrow{-1} ... \xrightarrow{-1} x_{k+1}$$

- x_2:

$$\pi[x_0, x_2] = \min\{\sigma_2, \pi_4, \pi_{15}\}$$

where

$$\pi_{15} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-k} x_{-k-1} \xrightarrow{-k} x_{-2k-1} \xrightarrow{-k} x_{-3k-1} \xrightarrow{-1} ... \xrightarrow{-1} x_2$$

Notice here that if $n = 4k + 1$, the agent cannot take route π_{15} because node $x_{-3k-1} = x_k$.

3. $\lceil \frac{n}{k} \rceil < 4$

- x_{k+1} In this case, in addition to the path σ_{k+1}, there are the following possibilities leading to:

$$\pi[x_0, x_{k+1}] = \min\{\sigma_{k+1}, \pi_{16}, \pi_{17}\}$$

  * (* The $+k^{th}$ neighbour of x_{k+1} is between x_{-1} and x_{-k}, so if x_{-1} is closer then: *)

$$\pi_{16} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-1} x_{-2} \xrightarrow{-1} ... \xrightarrow{-1} x_{2k+1} \xrightarrow{-k} x_{k+1}$$

* (* x_{-k} is closer to x_{k+1} *)

$$\pi_{17} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-k} x_{-k-1} \xrightarrow{-1} ... \xrightarrow{-1} x_{k+1}$$

- x_{2k}

$$\pi[x_0, x_{2k}] = \min\{\sigma_{2k}, \pi_{18}\}$$

$$\pi_{18} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-1} x_{-2} \xrightarrow{-1} ... \xrightarrow{-1} x_{2k}$$

- x_2

$$\pi[x_0, x_2] = \min\{\sigma_2, \pi_4, \pi_{15}\}$$

We have seen that the case of the shortly-chorded ring is simpler to study, however, the number of moves employed to reach the targets is the same in both cases and leads to the same complexity.

Theorem 7. *In any double loop $C_n(1, k)$, if $|S_{area}| \geq k$, the number of moves to complete the* Surrounding and Eliminating *phase is a maximum of* 20.

Proof. In any double loop chordal ring, if $|S_{area}| \geq k$, the maximum number of moves required to reach all targets is 20. One move is done by LEA to reach x_0. By construction, whether $\lceil \frac{n}{k} \rceil > 4$ or $\lceil \frac{n}{k} \rceil \leq 4$, the nodes are reached as follows: node x_{2k} is reached within four moves; node x_{k+1} is reached within six moves; node x_{k-1} is reached within two moves and node x_2 is reached within four moves.

Since we always compare the resulting routes π_z, where $z \in \mathbb{Z}$, to the special ones in the shortly-chorded loops σ_i, none of π_z would be greater than the corresponding σ_i.

After all the agents have arrived at their destinations, one move is done by the SH and two moves are needed to send CAs. □

Theorem 8. *In any double loop $C_n(1, k)$, if $|S_{area}| < k$, the number of moves to complete the Surrounding and Eliminating phase is a maximum of 36.*

Proof. At the beginning of the second phase, one move is made by the LEA to move to x_0. Then, whether $\lceil \frac{n}{k} \rceil > 4$ or $\lceil \frac{n}{k} \rceil \leq 4$, the nodes are reached as follows: node x_{2k} is reached within four moves; node x_{k+1} is reached within six moves and node x_{k-1} is reached within two moves. Node x_2 is reached by taking any route except the one that passes through x_{-k}, and the number of moves would be a maximum of eight. Node x_{-k-1} is reached within four moves; node x_{-k+1} is reached within six moves and node x_{-2k} is reached within four moves.

Since we always compare the resulting routes π_z, where $z \in \mathbb{Z}$, to the ones in the shortly-chorded loops σ_i, none of π_z would be greater than the corresponding σ_i.

After all agents arrive at their destinations, only three moves are needed to disinfect the topology. $\square$

The following table summarizes the number of moves required in the *Surrounding and Eliminating* phase in any double loop chordal ring.

Destination	Number of moves	
	$\lvert S_{area}\rvert < k$	$\lvert S_{area}\rvert \geq k$
x_0	1	1
x_2	≤ 8	≤ 4
x_{k-1}	2	2
x_{k+1}	≤ 6	≤ 6
x_{2k}	≤ 4	≤ 4
x_{-k-1}	2	-
x_{-2k}	4	-
x_{-k+1}	6	1
x_1	1	1
x_k	1	1
x_{-k}	1	-
Total	≤ 36	≤ 20

Table 4.1: Move complexity in double loops.

Theorem 9. *The algorithm following the move-optimal deployment strategy successfully disinfects a double loop chordal ring from black viruses in a monotone synchronous way.*

Proof. Destroying a *black virus* is done only if it moves to a guarded node. Therefore, in this phase, agents surround all the neighbouring nodes and then LEA activates them so they move to neighbouring nodes. As previously mentioned, the crucial part is the routing, and in this strategy (Move-optimal), LEA is responsible for directing agents to their destinations. Since LEA has the topological knowledge, it correctly calculates the targets and sets up the paths for SAs. $\square$

4.2.1.3 On Optimality and Other Observations

We ran a simulation to construct a partial *Breadth-First Search* tree rooted in x_0 in double loops with "missing nodes" corresponding to the black viruses triggered in the various scenarios. The spanning tree was constructed until all the targets appeared as leaves. In this type of spanning tree, any path from x_0 to a target leaf is the shortest path from x_0 to that destination.

We exhaustively tested all the different scenarios depending on the location of the black virus. Note that the size of the chordal ring influences the length of the shortest path for each scenario: 1) Shortly-chorded double loops when $|S_{area}| < k$, 2) Shortly-chorded double loops when $|S_{area}| \geq k$, 3) General double loops where $n \leq 4k$ when $|S_{area}| < k$ and 4) General double loops where $n \leq 4k$ when $|S_{area}| \geq k$. In all the scenarios above we verified that the paths indicated are indeed the shortest paths.

As a final note, if all agents have full knowledge of the topology and of the targets, the aforementioned approach can be transformed into a *local strategy* where agents decide their next step without referring back to the LEA. In each step, an agent could locally construct a breadth first search tree in order to find the shortest path to their target. They could then pick the neighbour that satisfies the following conditions: not a BV, not a predecessor and on the shortest path. After all the agents reach their destinations, LEA triggers the BVs by sending the cleaning agents. This strategy is optimal and local, yet it is compute-intensive. It requires the same number of agents and moves as the aforementioned move-optimal algorithm.

4.2.2 Simple Greedy Deployment

In the previous section, the proposed algorithm is controlled by the LEA who has a map for the entire topology. Once he finds the original BV, he decides the best routes and sends agents through them. The only job the agents have to do is following the paths set by the leader and stored in their memories. In this section we introduce some strategies that rely completely on local decisions. In other words, instead of carrying the whole path, the agent calculates its next move for each step until it reaches its target.

Consider the following simple local Greedy strategy: an agent at some node *source*, having to reach a node *dest*, chooses as a next link the one that takes it closest to *dest*. The distance is calculated on the outside ring. Note that, in a double loop *without viruses*, this simple greedy strategy would correspond to optimal routing between any pair of nodes.

Let $link(source, dest)$ be the next link to be taken from node *source* to eventually reach node *dest*. It is easy to see that the greedy strategy described above corresponds to the following local choice:

$$link(x_i, x_j) = \begin{cases} \text{if } (i-j) \leq (n-j+1) & go-clockwise: \\ \qquad \text{if } (j-i) < k & \text{take-link } +1 \\ \qquad \text{otherwise} & \text{take-link } +k \\ & \\ \text{if } (i-j) > (n-j+1) & go-counter-clockwise: \\ \qquad \text{if } (n-j+i) < k & \text{take-link -1} \\ \qquad \text{otherwise} & \text{take-link } -k \end{cases}$$

In order to occupy the target nodes, we have the following optimal paths where the BV has been underlined:

Target	Greedy path
x_2	$[x_0,\ \underline{x_1}, x_2]$
x_{k-1}	$[x_0,\ \underline{x_k}, x_{k-1}]$
x_{k+1}	$[x_0,\ \underline{x_k}, x_{k+1}]$
x_{2k}	$[x_0,\ \underline{x_k}, x_{2k}]$

Table 4.2: Greedy path without *black viruses.*

In the presence of BVs to be avoided, the greedy strategy will give correct but non-optimal routing. The advantage of this approach is that the surrounding agents do not carry a pre-determined path and movement decisions are made locally.

The greedy strategy can be described as follows. $dist(x, y)$ represents the shortest distance between nodes x and y along the ring.

```
SIMPLE GREEDY

Deploying agent $A$ arriving at $v$ from $y$ with destination $z$


    Let $FD = \{N(v) - \mathcal{BV} - y\}$ be the set of possible destinations.
    Agent $A$ moves to $w \in FD$ that minimizes $dist(w, z)$
```

Note that the first step in any surrounding strategy is always counter-clockwise, regardless of the destination, because both clockwise neighbours of x_0 are BVs. For example, an agent wants to reach x_2 from x_0 when $k > 5$. The agent would move to x_{-1} and would continue to move counter-clockwise until reaching node x_{-i} such that $i + 3 > -i + k - 2$

($2i > k-5$, $i = \lceil \frac{k-5}{2} \rceil$). At this point the agent would take the clockwise chord $+k$ which gets it closer to x_2, and then would proceed counter-clockwise towards it by using chords -1. Notice that when $k \leq 5$, i=1.

All the resulting greedy paths when $|S_{area}| \geq k$ are outlined in the table below. For simplicity, we assume that $5 < k < \lfloor \frac{n}{2} \rfloor$ and $k << n$.

Target	Greedy path with BVs	Number of moves
x_2	$[x_0, x_{-1}, x_{-2}, \ldots, x_{-i}, x_{-i+k}, x_{-i+k-1}, \ldots\ldots x_2]$ $i = \lceil \frac{k-5}{2} \rceil$	$k-1$
x_{k-1}	$[x_0, x_{-1}, x_{k-1}]$	2
x_{k+1}	$[x_0, x_{-1}, x_{k-1}, x_{k-2}, x_{k-3}, \ldots . x_{k-i}, x_{2k-i}, x_{2k-i-1}, \ldots, x_{k+1}]$ $i = \lceil \frac{k-3}{2} \rceil$	$k+1$
x_{2k}	$[x_0, x_{-1}, x_{k-1}, x_{2k-1}, x_{2k}]$	4

Table 4.3: Greedy paths with *black viruses* when $|S_{area}| \geq k$

Theorem 10. *In any double loop $C_n(1, k)$, if $|S_{area}| \geq k$ and $k > 5$, to surround and eliminate the black viruses, the total number of moves performed by the Simple Greedy strategy is $\leq 2k + 10$.*

Proof. The first move in this phase is performed by LEA in order to move from x_{-1} to x_0, then the routing begins. When $|S_{area}| \geq k$, after activating the original *black virus*, we might have two *black viruses*, one *black virus* or none depending on where the original *black virus* resides. If we have two *black viruses*, $\mathcal{T} = \{x_2, x_{k-1}, x_{k+1}, x_{2k}\}$, and the agents reach them as in table 4.3 and the number of moves is obtained as follows: node x_2 is reached in $(k-1)$ moves; node x_{k-1} is reached in two moves; node x_{2k} is reached in four moves; node x_{k+1} is reached in $(k+1)$ moves. The SH at x_{-k} makes one move to occupy x_{-k+1}, and LEA sends two CAs in two moves to disinfect the whole topology.

□

All the greedy paths when $|S_{area}| < k$ are outlined in the following table.

Target	Greedy path with BVs	Number of moves
x_2	$[x_0, x_{-1}, x_{-2}, \dots, x_{-i}, x_{-i+k}, x_{-i+k-1}, \dots\dots x_2]$ $i = \lceil \frac{k-5}{2} \rceil$	$k-1$
x_{k-1}	$[x_0, x_{-1}, x_{k-1}]$	2
x_{k+1}	$[x_0, x_{-1}, x_{k-1}, x_{k-2}, x_{k-3}, \dots x_{k-i}, x_{2k-i}, x_{2k-i-1}, \dots, x_{k+1}]$ $i = \lceil \frac{k-3}{2} \rceil$	$k+1$
x_{2k}	$[x_0, x_{-1}, x_{k-1}, x_{2k-1}, x_{2k}]$	4
x_{-k-1}	$[x_0, x_{-1}, x_{-k-1}]$	2
x_{-k+1}	$[x_0, x_{-1}, x_{-k-1}, x_{-k-2}, \dots x_{-k-i}, x_{-i}, x_{-i-1}, x_{-i-2}, \dots, x_{-k+1}]$ $i = \lceil \frac{k-3}{2} \rceil$	$k+1$
x_{-2k}	$[x_0, x_{-1}, x_{-k-1}, x_{-2k-1}, x_{-2k}]$	4

Table 4.4: Greedy paths with *black viruses* when $|S_{area}| < k$

As mentioned above, the greedy algorithm does not produce optimal paths. For example, consider the case $C_n\{1, 6\}$ with $\lceil \frac{n}{k} \rceil > 4$. To reach x_2 using the strategy described in the previous section requires four moves while in the greedy strategy the agent requires five moves.

Theorem 11. *In any double loop $C_n(1, k)$, if $|S_{area}| < k$ and $k > 5$, to surround and eliminate the black viruses, the total number of moves performed by the Simple Greedy strategy is $3k + 17$.*

Proof. The first move in this phase is performed by LEA to move from x_{-1} to x_0, then the routing begins. When $|S_{area}| < k$, after activating the original *black virus*, we have three *black viruses*. Thus $\mathcal{T} = \{x_2, x_{k-1}, x_{k+1}, x_{2k}, x_{-k-1}, x_{-k+1}, x_{-2k}\}$ and the agents reach them as in table 4.4 as follows: node x_2 is reached in $(k-1)$ moves; node x_{k-1} is reached in two moves; node x_{2k} is reached in four moves; node x_{k+1} is reached in $(k+1)$ moves; node x_{-k-1} is reached in two moves; node x_{-2k} is reached in four moves; node x_{-k+1} is reached in $(k+1)$ moves. The LEA then sends three CAs in three moves to disinfect the whole topology. $\square$

4.2.3 Smart Greedy Deployment

This algorithm is similar to the greedy algorithm except that this one takes the k^{th} chords into consideration.

```
SMART GREEDY

Deploying agent $A$ arriving at $v$ from $y$ with destination $z$


Agent $A$ calculates $next$.
    Let $FD = \{N(v) - \mathcal{BV} - y\}$ be the set of possible destinations.
    For each $v_i \in N(v)$
        Let $d_i = dist(v_i, z)$
        If $d_i \geq k$
        $d_i = \lfloor \frac{d_i}{k} \rfloor + (d_i \mod k)$
$next \in FD$ that has $v_i$ with the minimum $d_i$
    If there is a tie between two or more of $N(v)$
        TIE-BREAK.
Agent $A$ moves to $next$
```

```
TIE-BREAK

    $v_i, v_j \in N(v)$ where $v_i, v_j \notin BV$ and $dist(v_i, z) = dist(v_j, z)$
Agent $A$ calculates $next$
    Check $N(v_i)$ and $N(v_j)$
        For each $u \in N(v_i)$ and $w \in N(v_j)$ where $u, w \notin N(v)$ — are not predecessors
            Let $u_i = dist(v_i, z)$ and $w_j = dist(v_j, z)$
            for each $u_i$ or $w_j$ that is $\geq k$
                change the distance to the value $\lfloor \frac{dist}{k} \rfloor + (dist \mod k)$
    $next \in N(v)$ the has the neighbour with the minimum distance.
```

The smart greedy algorithm is local, yet it requires agents to do more calculations than the simple greedy algorithm. The complexity of this algorithm is not optimal. For example, to reach x_{k+1} in a shortly-chorded double loop, the agent needs $> six$ moves.

In the smart greedy algorithm, we consider the longest chord which plays an important role in decreasing the distance between the two nodes. Therefore, if the $dist(x, y) \geq k$, there is one or more long chords between x, y which minimizes the distance significantly. For example, if $dist(x, y) = 10$ in $C_n(1, 7)$, the actual distance is not 10 but 4 as we can see in the following equation: $dist(x, y) = \lfloor \frac{10}{7} \rfloor + (10 \mod 7) = 4$. Also, in this strategy, we attempted to resolve the situation in which two neighbours have the same distance to a target. If there is a tie, before the SA chooses, it checks the neighbours and finds the minimum among them. If the tie persists, the SA checks the neighbours of the neighbours until it breaks the tie. However, if the SA finds two neighbours have a common neighbour that gives the minimum distance, it can pick any of them as in the case of reaching x_{k+1} as we will see.

All the resulting smart greedy paths when $|S_{area}| \geq k$ are outlined in the table below. For simplicity, we assume that $5 < k < \lfloor \frac{n}{2} \rfloor$ and $k << n$.

Target	Smart Greedy path with BVs	Number of moves
x_2	$[x_0, x_{-k}, x_{-k}, x_{-k+1}, x_{-k+2}, x_2]$	4
x_{k-1}	$[x_0, x_{-1}, x_{k-1}]$	2
x_{k+1}	$[x_0, x_{-1}, x_{k-1}, x_{k-2}, ...x_{k-i}, x_{2k-i}, x_{2k-i-1}, \dots, x_{k+1}]$ $i = \lceil \frac{k-3}{2} \rceil$	$k+1$
x_{2k}	$[x_0, x_{-1}, x_{k-1}, x_{2k-1}, x_{2k}]$	4

Table 4.5: Smart greedy paths with *black viruses* when $|S_{area}| \geq k$

Theorem 12. *In any double loop $C_n(1, k)$ where $|S_{area}| \geq k$ and $k > 5$, to surround and eliminate the black viruses, the total number of moves performed by the Smart Greedy strategy is $\leq k + 15$.*

Proof. The first move in this phase is performed by LEA to move from x_{-1} to x_0, then the routing begins. When $|S_{area}| \geq k$, after activating the original *black virus*, we might have two *black viruses*, one *black virus* or none depending on where the original *black virus* resides. If we have two *black viruses*, $\mathcal{T}= \{x_2, x_{k-1}, x_{k+1}, x_{2k}\}$ and the agents reach them as in table 4.5 as follows: node x_2 is reached in four moves; node x_{k-1} is reached in two moves; node x_{2k} is reached in four moves; node x_{k+1} is reached in $(k + 1)$ moves. The SH at x_{-k} makes one move to occupy x_{-k+1}, and LEA sends two CAs in two moves to disinfect the whole topology. □

When $|S_{area}| < k$, all the resulting greedy paths are outlined in the table below.

Target	Smart Greedy path with BVs	Number of moves
x_2	$[x_0, x_{-1}, x_{-2}, \dots, x_{-i}, x_{-i+k}, x_{-i+k-1}, \dots \dots x_2]$ $i = \lceil \frac{k-5}{2} \rceil$	$k-1$
x_{k-1}	$[x_0, x_{-1}, x_{k-1}]$	2
x_{k+1}	$[x_0, x_{-1}, x_{k-1}, x_{k-2}, x_{k-3}, \dots . x_{k-i}, x_{2k-i}, x_{2k-i-1}, \dots, x_{k+1}]$ $i = \lceil \frac{k-3}{2} \rceil$	$k+1$
x_{2k}	$[x_0, x_{-1}, x_{k-1}, x_{2k-1}, x_{2k}]$	4
x_{-k-1}	$[x_0, x_{-1}, x_{-k-1}]$	2
x_{-k+1}	$[x_0, x_{-1}, x_{-k-1}, x_{-2k-1}, x_{-2k}, x_{-2k+1}, x_{-k+1}]$	6
x_{-2k}	$[x_0, x_{-1}, x_{-k-1}, x_{-2k-1}, x_{-2k}]$	4

Table 4.6: Smart greedy paths with *black viruses* when $|S_{area}| < k$

Theorem 13. *In any double loop $C_n(1,k)$, if $|S_{area}| < k$ and $k > 5$, to surround and eliminate the black viruses, the total number of moves performed by the Smart Greedy strategy is $2k + 22$.*

Proof. The first move in this phase is performed by LEA to move from x_{-1} to x_0, then the routing begins. When $|S_{area}| < k$, after activating the original *black virus*, we have three *black viruses*. Therefore, $\mathcal{T} = \{x_2, x_{k-1}, x_{k+1}, x_{2k}, x_{-k-1}, x_{-k+1}, x_{-2k}\}$ and the agents reach them as in 4.6 as follows: node x_2 is reached in $(k-1)$ moves; node x_{k-1} is reached in two moves; node x_{2k} is reached in four moves; node x_{k+1} is reached in $(k+1)$ moves; node x_{-k-1} is reached in two moves; node x_{-2k} is reached in four moves; node x_{-k+1} is reached in six moves. The LEA then sends three CAs in three moves to disinfect the whole topology. □

4.3 Conclusion

In this chapter we investigated the two phases required to disinfect a double loop chordal ring from *black viruses*. The first phase is common to all deployment strategies. In fact, as we will see throughout this thesis, this phase uses the same technique for all chordal rings. On the other hand, the second phase of double loop chordal rings has different possible routing strategies: move-optimal, simple greedy and smart greedy. The whole algorithm correctly disinfects any double loop chordal ring, regardless of the routing strategy.

In the following tables we combine the complexities of the two phases for the three routing strategies proposed for disinfecting any double loop chordal ring.

BV at node v_i	Phase 1	Move-Optimal		Simple Greedy		Smart Greedy	
		Agents	Moves	Agents	Moves	Agents	Moves
$1 \leq i < k$	$\leq 3k - 5$	12	≤ 36	12	$\leq 3k + 17$	12	$\leq 2k + 22$
$k \leq i < n - k$	$\leq 4n - 5k - 6$	9	≤ 20	9	$\leq 2k + 10$	9	$\leq k + 15$
$n - k \leq i < n - 1$	$\leq 4n - 12$	6	≤ 8	6	$k + 3$	6	≤ 8
$i = n - 1$	$\leq 4n - 7$	5	0	5	0	5	0

Table 4.7: Comparison between the complexities of the three strategies.

BV at node v_i	Phase1+ Move-Optimal	Phase1+ Simple Greedy	Phase1+Smart Greedy
$1 \leq i < k$	$\leq 3k + 31$	$\leq 6k + 12$	$\leq 5k + 17$
$k \leq i < n - k$	$\leq 4n - 5k + 14$	$\leq 4n - 3k + 4$	$\leq 4n - 4k + 9$
$n - k \leq i < n - 1$	$\leq 4n - 4$	$4n + k - 9$	$\leq 4n - 4$
$i = n - 1$	$4n - 7$	$4n - 7$	$4n - 7$

Table 4.8: The overall move complexity of the two phases in the three strategies.

By examining the tables above we can see that the complexity of the first phase remains the same regardless of the strategy chosen in the second phase. Moreover, the number of agents is the same for all of the strategies, while the number of moves varies. We can also see that when the *black virus* is found at node v_{n-1}, no moves are necessary in the *Surrounding and Eliminating* phase.

Even though we have only considered the problem in a synchronous setting, the algorithm would work in an asynchronous environment with some modifications. In order to make it work we would need an extra process for the termination phase which can be conducted by the LEA. After the LEA sends agents to their destinations, it will then move around and visit each site to confirm the successful arrival of each agent. This process would only add a constant number of moves.

Chapter 5

Black Virus Disinfection in Triple Loops

In this chapter, the process of locating and clearing the *Black Virus* using agents is considered in the shortly chorded *triple loop* chordal ring $C_n(1, p, k)$, where $1 < p << k$ and $k << n$, in a *synchronous* environment. Thee *triple loop* is a ring of n nodes $v_0, \ldots v_{n-1}$, where each node is connected to four additional neighbours at a distance of p and k. Thus, the neighbourhood of node v_i is obtained using: $N(v_i) = \{v_{i\pm1}, v_{i\pm p}, v_{i\pm k}\}$.

The exploration phase used to detect the black virus is the same as the one described for the double loop. We will see in this chapter that the surrounding phase, cannot be done using a local greedy algorithm for triple loops. Instead, we propose the use of a non-local move-optimal algorithm directed by the leader. Because of the complicated nature of exhaustively considering all possible combinations of p an k in the general case, we only provided the upper bounds of the optimal path lengths for the surrounding phase. We did however provide exact bounds for special triple loops corresponding to two extreme situations: $C_n(1, 2, k)$ and $C_n(1, k-1, k)$.

5.1 Triple Loops

In this section we discuss triple loop chordal rings $C_n(1, p, k)$ with arbitrary numbers of p, k and $k << n$.

5.1.1 Exploring and Shadowing

As previously explained in the double loop section, this phase is fixed regardless of the surrounding method or the structure of the chords. The LEA and EA agents explore each node on the outer ring in a clockwise fashion using the *safe exploration* technique. In order to achieve monotonicity and diminish the effect of the BV, shadow agents are deployed to guard the explored neighbours of the next node to be visited. This cannot happen unless LEA and EA have passed through at least p nodes. In other words, if the safe area is $\geq p$.

The correctness of this phase stems from the fact that the exploring team follows the outer ring and that EA always precedes LEA, so that the *black virus* is eventually detected in a monotone fashion. As we discussed in 1, monotonicity is achieved since the maximum number of SHs recruited from the beginning is five (i.e., as many as the number of neighbours in the worst case $v = n - 1$).

```
EXPLORING AND SHADOWING

let $HB = v_0$
Agents $EA$ and LEA at safe node $v_i$.

if $(1 \leq i+1 < p)$
    $N_{ex}(v_{i+1}) = \{v_i\}$
    no $SH$ is deployed
if $(p \leq i+1 < k)$
    $N_{ex}(v_{i+1}) = \{v_i, v_{i+1-p}\}$
    1 $SH$ is deployed to protect $v_{i+1-p}$
if $(k \leq i+1 < n-k)$
    $N_{ex}(v_{i+1}) = \{v_i, v_{i+1-p}, v_{i+1-k}\}$
    2 $SH$s are deployed to protect $v_{i+1-p}, v_{i+1-k}$
if $(n-k \leq i+1 < n-p)$
    $N_{ex}(v_{i+1}) = \{v_i, v_{i+1-p}, v_{i+1-k}, v_{i+1+k}\}$
    3 $SH$s are deployed to protect $v_{i+1-p}, v_{i+1-k}, v_{i+1+k}$
if $(n-p \leq i+1 < n-1)$
    $N_{ex}(v_{i+1}) = \{v_i, v_{i+1-p}, v_{i+1-k}, v_{i+1+k}, v_{i+1+p}\}$
    4 $SH$s are deployed to protect $v_{i+1-p}, v_{i+1-k}, v_{i+1+k}, v_{i+1+p}$
if $(i = n-2)$
    $N_{ex}(v_{i+1}) = \{v_i, v_{i+1-p}, v_{i+1-k}, v_{i+1+k}, v_{i+1+p}, v_0\}$
    5 $SH$s are deployed to protect $v_{i+1-p}, v_{i+1-k}, v_{i+1+k}, v_{i+1+p}, v_0$

EA moves to $v_{i+1}$.
```

After employing this strategy we made the following observations:

Theorem 14. *In the worst case scenario, the black virus is detected in* $5n - 9$ *moves.*

Proof. The worst case for the number of moves required occurs when the *black virus* is found at node v_i, where $i = n - 1$. In this case, the BV triggers no new *black viruses* since all the neighbours have already been explored in the safe area and are all protected by SHs. The complexity of this case would be $(3(n-1)-2)$ for the movement of LEA and EA, $(n-1-k)$ for one SH, $(n-1-p)$ for the second SH, $(k-1)$ for the third SH and $(p-1)$ for the fourth SH for a total $5n-9$ moves. □

Theorem 15. *In any triple loop chordal ring C, the worst case scenario in term of the number of agents required occurs when five new black viruses are created after triggering the original virus.*

Proof. The worst case for the number of cleaning agents (CA) and surrounding agents (SA) occurs when the *black virus* is located at node v_i where $1 \leq i < p$. In this case, where the *black virus* is x_0, it triggers five new *black viruses* at x_1, x_p, x_k, x_{-p} and x_{-k} because no SH have been deployed. x_{-1} is always occupied by LEA. □

5.1.2 Surrounding and Eliminating

In the triple loop chordal ring, when the *black virus* is triggered it may affect up to five neighbouring nodes. If x_0 is the original BV node,, node x_{-1} is always protected by LEA. x_1,x_p, x_k,x_{-p} and x_{-k} are protected only if they belong to the safe area and are occupied by SHs. To summarize, the best case scenario in terms of the number of *black viruses*, after EA triggers the original black virus, is that no more BVs are created, while the worst case scenario is that five BVs are created.

In order to handle the spread of BVs, the LEA has to send agents to surround and clear those faults. As with double loops, the LEA creates SA(s) to be sent to specific targets, and when all agents are in position, the LEA sends CA(s) to trigger all the BVs at the same time. Thus, we need as many SAs as the neighbours of the BVs and as many CAs as the BVs.

```
SURROUNDING AND ELIMINATING

LEA and $SH$s covering all $N_{ex}(v)$
$BV$ comes back from $v = x_0$.


if ($|S_{area}| < p$) (*LEA is covering $x_{-1}$*)
    $N_{un}(x_0) = \{x_1, x_p, x_k, x_{-p}, x_{-k}\}$
if ($p \leq |S_{area}| < k$) (*LEA and $SH$ covering $x_{-1}, x_{-p}$*)
    $N_{un}(x_0) = \{x_1, x_k, x_p, x_{-k}\}$
if ($k \leq |S_{area}| < n - k$) (*LEA and 2$SH$s covering $x_{-1}, x_{-p}, x_{-k}$*)
    $N_{un}(x_0) = \{x_1, x_p, x_k\}$
if ($n - k \leq |S_{area}| < n - p$) (*LEA and $SH$s covering $x_{-1}, x_{-p}, x_{-k}, x_k$*)
    $N_{un}(x_0) = \{x_1, x_p, \}$
if ($n - p \leq |S_{area}| < n - 1$)(*LEA and $SH$s covering $x_{-1}, x_{-p}, x_{-k}, x_k, x_p$*)
    $N_{un}(x_0) = \{x_1\}$
Else (*LEA and $SH$s covering $x_{-1}, x_{-p}, x_{-k}, x_k, x_p, x_1$*)
    $N_{un}(x_0) = \emptyset$


All $SH$ make one move in the clockwise direction.
For each $u \in N_{un}(x_0)$:
    DEPLOY an agent to each $z \in \{N(u) \setminus N_{un}(x_0)\}$
When $N(u)$ is covered:
    DEPLOY one agent to $u$
```

For our convenience, we separated the topology into five segments, where the *black virus* at (v_i) is located in one of those five segments as seen in figure 5.1.

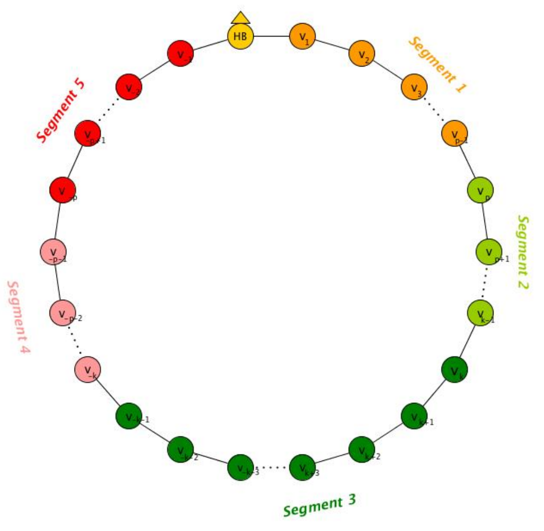

Figure 5.1: Dividing the triple loop into five segments.

- **Segment 1:** this segment contains nodes $v_1, v_2, ..., v_{p-1}$ and the size of the safe area is $|S_{area}| \leq p-1$.
- **Segment 2:** this segment contains nodes $v_p, v_{p+1}, ..., v_{k-1}$ and the size of the safe area is $p \leq |S_{area}| \leq k-1$.
- **Segment 3:** this segment contains nodes $v_k, v_{k+1}, ..., v_{n-k-1}$ and the size of the safe area is $k \leq |S_{area}| \leq n-k-1$.
- **Segment 4:** this segment contains nodes $v_{n-k}, v_{n-k+1}, ..., v_{n-p-1}$ and the size of the safe area is $n-k \leq |S_{area}| \leq n-p-1$.
- **Segment 5:** this segment contains nodes $v_{n-p}, v_{n-p+1}, ..., v_{n-1}$ and the size of the safe area is $n-p \leq |S_{area}| \leq n-1$.

The number of agents required to disinfect the triple loop chordal ring $C_n(1, p, k)$ is fixed regardless of the surrounding strategy employed, whereas the number of moves necessary varies depending on the deployment method.

Theorem 16. *Regardless of deployment strategy and chords length, a maximum of* 24 *agents are employed in any triple loop $C_n(1, p, k)$ for black virus disinfection.*

Proof. The number of agents is determined by the location of the original BV, regardless of the deployment method. As previously mentioned, the *black virus* is found in one of five segments:

- If the BV is located at any node in *Segment 1*, we would have the worst complexity in terms of CA and SA since activating the *black virus* will create five more *black viruses* at x_1, x_p, x_k, x_{-p} and x_{-k}. In addition to the LEA which is located at x_0, the LEA deploys 17 SAs to occupy x_2, x_{p-1}, x_{k-1}, x_{p+1}, x_{k+1}, x_{k-p}, x_{k+p}, x_{2p}, x_{2k}, x_{-k+p}, x_{-p+1}, x_{-p-1}, x_{-k+1}, x_{-k-1}, x_{-k-p}, x_{-2p} and x_{-2k}. If the BV is found in this segment, $Spread(C) = 6$ and $Size(C) = 24$.

- If the BV is located at any node in *Segment 2*, activating the original *black virus* would create four more *black viruses* at x_1, x_p, x_k and x_{-k} while x_{-p} is guarded by a SH. In addition to the SH at x_{-p+1} and LEA at x_{-1}, the LEA deploys 14 SAs to occupy x_2, x_{p-1}, x_{k-1}, x_{p+1}, x_{k+1}, x_{k-p}, x_{k+p}, x_{2p}, x_{2k}, x_{-k+p}, x_{-k+1}, x_{-k-1}, x_{-k-p} andx_{-2k}. If BV is found in this segment, $Spread(C) = 5$ and $Size(C) = 21$.

- If the BV is located at any node in *Segment 3*, activating the original *black virus* would create three more *black viruses* at x_1, x_p and x_k while x_{-p} and x_{-k} are guarded by SHs. Therefore, in addition to the SHs at x_{-p+1}, x_{-k+1} and LEA at x_0,the LEA deploys 10 SAs to occupy x_2, x_{p-1}, x_{k-1}, x_{p+1}, x_{k+1}, x_{k-p}, x_{k+p}, x_{2p}, x_{2k} and x_{p-k}. If the BV is found in this segment, $Spread(C) = 4$ and $Size(C) = 17$.

- *Segment 4* is part of the so called *Danger area* (D_{area}). Since our protocol is monotone, the explored nodes should be protected from reinfection; therefore, more SH(s) are deployed. If the BV is located at any node in this segment, activating the original *black virus* would create two more *black viruses* at x_1 and x_p while x_{-p}, x_{-k} and x_k are guarded by SHs. Therefore, in addition to the SHs at $x_{-p+1}, x_{-k+1}, x_{k+1}$ and LEA at x_0, the LEA deploys 6 SAs to occupy x_2, x_{p-1}, x_{p+1}, x_{p-k}, x_{p+k} and x_{2p}. $Spread(C) = 3$ and $Size(C) = 13$.

- *Segment 5* is also a part of the *Danger area* (D_{area}) so more SHs are deployed in order to maintain the monotonicity of our protocol. This segment is divided into two cases:

  - When $n - p \leq i < n - 1$:

    If the BV is located at any node in this segment, activating the original *black virus* would create only one *black virus* at x_1, while x_{-p}, x_{-k}, x_k and x_p are guarded by SHs. Subsequently, in addition to the SH at $x_{-p+1}, x_{-k+1}, x_{k+1}, x_{p+1}$, and LEA at x_0, the LEA deploys 1 SA to occupy x_2. $Spread(C) = 2$ and $Size(C) = 8$.

  - When $i = n - 1$: In this case, no more *black viruses* are created since all neighbours are guarded by SHs and the LEA. No SAs are deployed and all the moves are done in the first phase. $Spread(C) = 1$ and $Size(C) = 7$.

□

5.1.2.1 Greedy Deployment.

We noted that the greedy strategy could not be used for the deployment part of this phase, because, according to the chord structure, an agent could be caught in an infinite loop. For example, consider a triple loop $C_n(1, 2, k > 9)$ with the target x_3. Starting from x_0, an agent would move greedily to x_{-1} then to x_{-2}; when the agent reached x_{-2} it would move to the closest neighbour of x_3, excluding x_{-1} and any infected neighbour, thus moving to x_0 again and forming an infinite loop.

As a result, we focus only on the non-local, move-optimal approach where optimal paths are computed by the LEA.

5.1.2.2 Move-Optimal Deployment.

Due to the complexity of the chord structure, we can only provide an upper bound on the length of the optimal paths.

After triggering the original BV, which is x_0, the neighbouring nodes are in one of two states: guarded or contaminated. Node x_{-1} is guarded by LEA. Nodes x_1, x_p, x_k, x_{-p} and x_{-k} could be in either state depending on the size of the *safe area*. All possible targets in the worst case scenario are: $\mathcal{T}$=$\{x_2$, x_{p-1}, x_{p-k}, x_{k-1}, x_{p+1}, x_{k+1}, x_{k-p}, x_{k+p}, x_{2p}, x_{2k}, x_{-p+1}, x_{-p-1}, x_{-k+1}, x_{-k-1}, x_{-k-p}, x_{-2p}, $x_{-2k}\}$. In fact, sometimes some of these targets could be BVs or common neighbours depending on the structure of the chordal ring.

We now specify a path for each target, the length of which is certainly an upper bound of the optimal path length. In this type of chordal ring, the length of these paths is constant, regardless of the target. As we did for the double loop, we have listed the special paths σ_i below.

Target x_i	Special Path σ_i
x_2	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k} x_{k-1} \xrightarrow{+k} x_{2k-1} \xrightarrow{+1} x_{2k} \xrightarrow{+1} x_{2k+1} \xrightarrow{+1} x_{2k+2} \xrightarrow{-k} x_{k+2} \xrightarrow{-k} x_2$
x_{p-1}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+p} x_{p-1}$
x_{k-1}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k} x_{k-1}$
x_{p-k}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+p} x_{p-1} \xrightarrow{-k} x_{-k+p-1} \xrightarrow{+1} x_{p-k}$
x_{p+1}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+p} x_{p-1} \xrightarrow{+p} x_{2p-1} \xrightarrow{+1} x_{2p} \xrightarrow{+1} x_{2p+1} \xrightarrow{-p} x_{p+1}$
x_{k+1}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k} x_{k-1} \xrightarrow{+k} x_{2k-1} \xrightarrow{+1} x_{2k} \xrightarrow{+1} x_{2k+1} \xrightarrow{-p} x_{k+1}$
x_{k-p}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k} x_{k-1} \xrightarrow{-p} x_{k-p-1} \xrightarrow{+1} x_{k-p}$
x_{k+p}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+p} x_{p-1} \xrightarrow{+k} x_{k+p-1} \xrightarrow{+1} x_{k+p}$
x_{2p}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+p} x_{p-1} \xrightarrow{+p} x_{2p-1} \xrightarrow{+1} x_{2p}$
x_{2k}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k} x_{k-1} \xrightarrow{+k} x_{2k-1} \xrightarrow{+1} x_{2k}$
x_{-p+1}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-p} x_{-p-1} \xrightarrow{-p} x_{-2p-1} \xrightarrow{+1} x_{-2p} \xrightarrow{+1} x_{-2p+1} \xrightarrow{+p} x_{-p+1}$
x_{-k+1}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-k} x_{-k-1} \xrightarrow{-k} x_{-2k-1} \xrightarrow{+1} x_{-2k} \xrightarrow{+1} x_{-2k+1} \xrightarrow{+k} x_{-k+1}$
x_{-p-1}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-p} x_{-p-1}$
x_{-k-1}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-k} x_{-k-1}$
x_{-k-p}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-k} x_{-k-1} \xrightarrow{-p} x_{-k-p-1} \xrightarrow{+1} x_{-k-p}$
x_{-2p}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-p} x_{-p-1} \xrightarrow{-p} x_{-2p-1} \xrightarrow{+1} x_{-2p}$
x_{-2k}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-k} x_{-k-1} \xrightarrow{-k} x_{-2k-1} \xrightarrow{+1} x_{-2k}$

Depending on the chords structure, shorter paths can be devised in some cases. A detailed analysis of all possible situations is carried out in Appendix A.

The following table summarizes the upper bound of the number of moves required to reach each possible target. Notice that we distinguish between two situations: when $|S_{area}| < p$ and when $|S_{area}| \geq k$. The former situation represents the worst possible number of *black viruses* while the latter situation combines different cases where the number of *black viruses* decreases according to the location of the original *black virus.*

Destination	Number of moves	
	$\lvert S_{area}\rvert < p$	$\lvert S_{area}\rvert \geq k$
x_0	1	1
x_2	≤ 8	≤ 4
x_{p-1}	2	2
x_{k-1}	2	2
x_{p+1}	≤ 6	≤ 4
x_{k+1}	≤ 6	≤ 6
x_{k-p}	≤ 4	≤ 4
x_{k+p}	≤ 4	≤ 4
x_{2p}	≤ 4	≤ 4
x_{2k}	4	4
x_{-p+1}	≤ 6	1
x_{-p-1}	2	-
x_{-2p}	4	-
x_{p-k}	≤ 4	≤ 4
x_{-k+1}	≤ 6	1
x_{-k-1}	2	-
x_{-k-p}	4	-
x_{-2k}	4	-
x_1	1	1
x_p	1	1
x_k	1	1
x_{-p}	1	-
x_{-k}	1	-
Total	≤ 78	≤ 44

Table 5.1: Move complexity in shortly-chorded triple loops.

Theorem 17. *In any triple loop $C_n(1, p, k)$ where $|S_{area}| \geq k$, the number of moves required to surround and eliminate BVs is ≤ 44.*

Proof. In any triple loop chordal ring where $|S_{area}| \geq k$, the maximum number of moves required to reach all targets is 44: One move is done by the LEA to reach x_0. Node x_2 is reached within four moves. Node x_{p-1} is reached within two moves. Node x_{k-1} is reached within two moves. Node x_{p+1} is reached within four moves. Node x_{k+1} is reached within six moves. Node x_{k-p} is reached within four moves. Node x_{k+p} is reached within four moves. Node x_{2p} is reached within four moves. Node x_{2k} is reached within four moves. Node x_{p-k} is reached within four moves. Two moves are made by the SHs in order to move from x_{-p} and x_{-k} to x_{-p+1} and x_{-k+1} when they receive copies of the original BV. After all of the agents arrive at their destinations the LEA sends three CAs to the *black viruses* at x_1, x_p and x_k in three moves. Notice that since we always compare the resulting routes π_z, where $z \in \mathbb{Z}$ to the σ_i, none of π_z would be greater than the corresponding σ_i. □

Theorem 18. *In any triple loop $C_n(1, p, k)$ where $|S_{area}| < p$, the number of moves required to surround and eliminate the BVs is ≤ 78.*

Proof. In any triple loop chordal ring where $|S_{area}| < p$, the maximum number of moves required to reach all targets is 78. One move is made by the LEA to reach x_0. Node x_2 is reached within eight moves. Node x_{p-1} is reached within two moves. Node x_{k-1} is reached within two moves. Node x_{p+1} is reached within six moves. Node x_{k+1} is reached within six moves. Node x_{k-p} is reached within four moves. Node x_{k+p} is reached within four moves. Node x_{2p} is reached within four moves. Node x_{2k} is reached within four moves. Node x_{-p+1} is reached within six moves. Node x_{-p-1} is reached within two moves. Node x_{-2p} is reached within four moves. Node x_{p-k} is reached within four moves. Node x_{-k+1} is reached within six moves. Node x_{-k-1} is reached within two moves. Node x_{-k-p} is reached within four moves. Node x_{-2k} is reached within four moves. After all agents arrive at their destinations the LEA sends five CAs to the *black viruses* at x_1, x_p, x_k,x_{-p} and x_{-k} in five moves. Notice that since we always compare the resulting routes π_z, where $z \in \mathbb{Z}$ to the σ_i, none of π_z would be greater than the corresponding σ_i. □

In the following two sections we will investigate the black virus disinfection problem in triple loop chordal rings for the two extreme cases previously mentioned. In these cases we can derive an exact bound on the optimal path lengths.

5.2 Triple Loops: the Case of $C_n(1, 2, k)$

5.2.1 Exploring and Shadowing

As we explained earlier, this phase is common to all chordal rings. We instantiate it below for the particular case of $C_n(1, 2, k)$.

```
EXPLORING AND SHADOWING

let $HB = v_0$
Agents $EA$ and LEA at safe node $v_i$.

if $(i = 0)$
    $N_{ex}(v_{i+1}) = \{v_0\}$
    no $SH$ has been deployed
if $(2 \leq i + 1 < k)$
    $N_{ex}(v_{i+1}) = \{v_{i-1}, v_i\}$
    1 $SH$ has been deployed to protect $v_{i-1}$
if $(k \leq i + 1 < n - k)$
    $N_{ex}(v_{i+1}) = \{v_i, v_{i-1}, v_{i+1-k}\}$
    2 $SH$s have been deployed to protect $v_{i-1}, v_{i+1-k}$
if $(n - k \leq i + 1 < n - 2)$
    $N_{ex}(v_{i+1}) = \{v_i, v_{i-1}, v_{i+1-k}, v_{i+1+k}\}$
    3 $SH$s have been deployed to protect $v_{i-1}, v_{i+1-k}, v_{i+1+k}$
if $(i = n - 3)$
    $N_{ex}(v_{i+1}) = \{v_i, v_{i-1}, v_{i+1-k}, v_{i+1+k}, v_0\}$
    4 $SH$s have been deployed to protect $v_{i-1}, v_{i+1-k}, v_{i+1+k}, v_0$
if $(i = n - 2)$
    $N_{ex}(v_{i+1}) = \{v_i, v_{i-1}, v_{i+1-k}, v_{i+1+k}, v_1, v_0\}$
    5 $SH$s have been deployed to protect $v_{i-1}, v_{i+1-k}, v_{i+1+k}, v_1, v_0$

EA moves to $v_{i+1}$.
```

The observations below were made following the employment of this strategy:

Theorem 19. *In the worst case scenario, the black virus is detected using* $5n - 9$ *moves.*

Proof. The worst case scenario for the number of moves required occurs when the *black virus* is found at node v_i, where $i = n - 1$. In this case the BV triggers no new *black viruses* since all of the neighbouring nodes in the safe area have been explored and all are protected by SHs. The complexity of this case would be $(3(n-1)-2)$ for the movement of LEA and EA, $(n-1-k)$ for one SH to guard node v_{n-1-k}, $(n-3)$ for the second SH to guard node v_{n-3}, $(k-1)$ for the third SH to guard node v_{k-1} and (1) for the fourth SH to guard node v_1 for a total of $5n - 9$ moves. □

Theorem 20. *In any triple loop chordal ring* $C_n = \{1, 2, k\}$*, the worst case scenario in terms of the number of agents required occurs when five new black viruses are created after triggering the original virus.*

Proof. The worst case scenario in terms of the number of cleaning agents (CAs) and surrounding agents (SAs) occurs when the *black virus* is at node v_i where $i = 1$. In this case, where the *black virus* is x_0, it triggers five new *black viruses* at x_1, x_2, x_k, x_{-2} and x_{-k} because no SH have been deployed. Note that x_{-1} is always occupied by LEA. In this case, the maximum number of *black viruses* would be created. □

5.2.2 Surrounding and Eliminating

As with the general case, in this class of triple loops, when the *black virus* is triggered it has the potential to affect up to five of its neighbours by the end of the first phase. In the second phase, the LEA creates SA(s) to be sent to specific targets and then sends CA(s) to activate all the *black viruses* at the same time and disinfect the entire topology.

```
SURROUNDING AND ELIMINATING

LEA and $SH$s covering all $N_{ex}(v)$
if $(|S_{area}| = 1)$( *LEA is covering $x_{-1}$*)
    $N_{un}(x_0) = \{x_1, x_2, x_k, x_{-2}, x_{-k}\}$
if $(2 \leq |S_{area}| < k)$(*LEA and $SH$ covering $x_{-1}, x_{-2}$*)
    $N_{un}(x_0) = \{x_1, x_2, x_k, x_{-k}\}$
if $(k \leq |S_{area}| < n - k)$(*LEA and 2 $SH$s covering $x_{-1}, x_{-2}, x_{-k}$*)
    $N_{un}(x_0) = \{x_1, x_2, x_k\}$
if $(n - k \leq |S_{area}| < n - 2)$(*LEA and $SH$s covering $x_{-1}, x_{-2}, x_{-k}, x_k$*)
    $N_{un}(x_0) = \{x_1, x_2, \}$
if $(|S_{area}| = n - 2)$(*LEA and $SH$s covering $x_{-1}, x_{-2}, x_{-k}, x_k, x_2$*)
    $N_{un}(x_0) = \{x_1\}$
Else (*LEA and $SH$s covering $x_{-1}, x_{-2}, x_{-k}, x_k, x_2, x_1$*)
    $N_{un}(x_0) = \emptyset$


All $SH$ make one move in the clockwise direction.
For each $u \in N_{un}(x_0)$:
    DEPLOY an agent to each $z \in \{N(u) \setminus N_{un}(x_0)\}$
When $N(u)$ is covered:
    DEPLOY one agent to $u$
```

The exploring team will find the *black virus* at (v_i), which is located in one of the five segments of the chordal ring:

- **Segment 1**: this segment contains node v_1 and the size of the safe area is $|S_{area}| = 1$.
- **Segment 2**: this segment contains nodes $v_2, v_3, ..., v_{k-1}$ and the size of the safe area is $2 \leq |S_{area}| \leq k - 1$.

- **Segment 3**: this segment contains nodes $v_k, v_{k+1}, ..., v_{n-k-1}$ and the size of the safe area is $k \leq |S_{area}| \leq n-k-1$.

- **Segment 4**: this segment contains nodes $v_{n-k}, v_{n-k+1}, ..., v_{n-3}$ and the size of the safe area is $n-k \leq |S_{area}| \leq n-3$.

- **Segment 5**: this segment contains nodes v_{n-2} and v_{n-1} and the size of the safe area is $n-2 \leq |S_{area}| \leq n-1$.

The number of agents required to disinfect the triple loop chordal ring($C_n(1,2,k)$) is fixed, regardless of the surrounding strategy, whereas the number of moves varies depending on the deployment method.

Theorem 21. *Regardless of the deployment strategy, a maximum of 22 agents are employed in any triple loop $C_n(1,2,k)$ for black virus disinfection.*

Proof. The number of agents required is determined by the location of the original BV, regardless of the deployment method.

- If the BV is found in *Segment 1* we would have the worst complexity in terms of CAs and SAs since activating the *black virus* will create five more *black viruses* at x_1, x_2, x_k, x_{-2} and x_{-k}. In addition to the LEA which is at x_0, the LEA deploys 15 SAs to occupy x_{-1}, x_3, x_4, x_{k-2}, x_{k-1}, x_{k+1}, x_{k+2}, x_{2k}, x_{-3}, x_{-4}, x_{-k+1}, x_{-k+2}, x_{-k-1}, x_{-k-2} and x_{-2k}. Therefore, $Spread(C) = 6$ and $Size(C) = 22$.

- If the BV is found in *Segment 2*, activating the original *black virus* would create four more *black viruses* at x_1, x_2, x_k and x_{-k} while x_{-2} is guarded by a SH. In addition to the SH at x_{-1} and LEA at x_0, the LEA deploys 12 SAs to occupy x_3, x_4, x_{k-2}, x_{k-1}, x_{k+1}, x_{k+2}, x_{2k}, x_{-k+1}, x_{-k+2}, x_{-k-1}, x_{-k-2} and x_{-2k}. Therefore, $Spread(C) = 5$ and $Size(C) = 19$.

- If the BV is found in *Segment 3*, activating the original *black virus* would create three more *black viruses* at x_1, x_2 and x_k while x_{-2} and x_{-k} are guarded by SHs. Therefore, in addition to the SHs at x_{-1} and x_{-k+1} and LEA at x_0, the LEA deploys 8 SAs to occupy x_3, x_4, x_{k-2}, x_{k-1}, x_{k+1}, x_{k+2}, x_{2k} and x_{-k+2}. Therefore, $Spread(C) = 4$ and $Size(C) = 15$.

- If the BV is found in *Segment 4*, activating the original *black virus* would create two more *black viruses* at x_1 and x_2 while x_{-2}, x_{-k} and x_k are guarded by SHs. Therefore, in addition to the SHs at x_{-1}, x_{-k+1} and x_{k+1} and LEA at x_0, the LEA deploys 4 SAs to occupy x_3, x_4 x_{-k+2} and x_{k+2}. Therefore, $Spread(C) = 3$ and $Size(C) = 11$.

- If the BV is found in *Segment 5*, we have two possible cases:

  - When $i = n - 2$:

    If the BV is located at node v_{n-2}, activating the original *black virus* would create only one *black virus* at x_1, while x_{-2}, x_{-k}, x_k and x_2 are guarded by SHs. Subsequently, in addition to the SH at $x_{-1}, x_{-k+1}, x_{k+1}, x_3$ and LEA at x_0, the LEA deploys 1 SA to occupy x_2. Therefore, $Spread(C) = 2$ and $Size(C) = 8$.

  - When $i = n - 1$:

    In this case, no more *black viruses* are created since all the neighbouring nodes are guarded by SHs and the LEA. No SAs need to be deployed and all the moves are made in the first phase. Therefore, $Spread(C) = 1$ and $Size(C) = 7$.

□

For the deployment part of this phase, we have demonstrated that a local greedy approach does not yield correct results, see 5.1.2.1. In the following section we discuss the non-local surrounding strategy in which SAs follow specific paths set up by the LEA which has knowledge of the topology.

5.2.2.1 Move-Optimal Deployment

As with the general triple loop, for the surrounding phase we specify each path for each target. Let us now consider all of the possible targets: $\mathcal{T}$=$\{x_{-1}$, x_3, x_4, x_{k-2}, x_{k-1}, x_{k+1}, x_{k+2}, x_{2k}, x_{-3}, x_{-4}, x_{-k+1}, x_{-k+2}, x_{-k-1}, x_{-k-2}, $x_{-2k}\}$. The special paths (σ_i) for this type of triple loop are described below.

Target x_i	Special Path σ_i
x_{-1}	$x_0 \xrightarrow{-1} x_{-1}$
x_{k-1}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k} x_{k-1}$
x_{k-2}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k} x_{k-1} \xrightarrow{-1} x_{k-2}$
x_{k+1}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k} x_{k-1} \xrightarrow{+2} x_{k+1}$
x_{k+2}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k} x_{k-1} \xrightarrow{+2} x_{k+1} \xrightarrow{+1} x_{k+2}$
x_3	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k} x_{k-1} \xrightarrow{+2} x_{k+1} \xrightarrow{+2} x_{k+3} \xrightarrow{-k} x_3$
x_4	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k} x_{k-1} \xrightarrow{+2} x_{k+1} \xrightarrow{+2} x_{k+3} \xrightarrow{+1} x_{k+4} \xrightarrow{-k} x_4$
x_{2k}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k} x_{k-1} \xrightarrow{+k} x_{2k-1} \xrightarrow{+1} x_{2k}$
x_{-3}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-2} x_{-3}$
x_{-4}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-2} x_{-3} \xrightarrow{-1} x_{-4}$
x_{-k-1}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-k} x_{-k-1}$
x_{-k-2}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-k} x_{-k-1} \xrightarrow{-1} x_{-k-2}$
x_{-k+1}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-k} x_{-k-1} \xrightarrow{+2} x_{-k+1}$
x_{-k+2}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-k} x_{-k-1} \xrightarrow{+2} x_{-k+1} \xrightarrow{+1} x_{-k+2}$
x_{-2k}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-k} x_{-k-1} \xrightarrow{-k} x_{-2k-1} \xrightarrow{+1} x_{-2k}$

Depending on the chords structure, shorter paths could be devised in some cases. A detailed analysis of all the possible situations is carried out in the following discussion.

Let us now consider the different routes to each target in six different cases, depending on the location of the *black virus* where $\pi[x_0, x_i]$ denotes a path to reach target x_i.

- **Case 1**: In this case we examine the process of finding the *black virus* in the third segment of the chordal ring: $k \le |S_{area}| < n-k$. In this case, triggering the original *black virus* creates three more *black viruses* : x_1, x_2 and x_k, and thus $\mathcal{T}$=$\{x_3$, x_4, x_{k-2}, x_{k-1}, x_{k+1}, x_{k+2}, x_{2k}, $x_{-k+2}\}$.

  - x_3 is reached as follows:

    $$\pi[x_0, x_3] = \min\{\pi_1, \pi_2\}$$

    Taking advantage of the fact that node $x_{-k} \in S_{area}$, node x_3 is reached through π_1 where

    $$\pi_1 = x_0 \xrightarrow{-k} x_{-k} \xrightarrow{+1} x_{-k+1} \xrightarrow{+2} x_{-k+3} \xrightarrow{+k} x_3$$

    or through π_2

    $$\pi_2 = x_0 \xrightarrow{-1} x_{k-1} \xrightarrow{-2} x_{k-3} \xrightarrow{-2} x_{k-5} \xrightarrow{-2}, ..., \xrightarrow{-i} x_3$$

    where $i = 1$ or $i = 2$.

  - x_4 is reached as follows:

    $$\pi[x_0, x_4] = \min\{\pi_3, \pi_4\}$$

    Taking advantage of the fact that node $x_{-k} \in S_{area}$, node x_4 is reached through π_3 where

    $$\pi_3 = x_0 \xrightarrow{-k} x_{-k} \xrightarrow{+2} x_{-k+2} \xrightarrow{+2} x_{-k+4} \xrightarrow{+k} x_4$$

or through π_4

$$\pi_4 = x_0 \xrightarrow{-1} x_{k-1} \xrightarrow{-2} x_{k-3} \xrightarrow{-2} x_{k-5} \xrightarrow{-2}, ..., \xrightarrow{-i} x_4$$

where $i = 1$ or $i = 2$.

– x_{k-1} is reached through σ_{k-1}

$$\sigma_{k-1} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k} x_{k-1}$$

– x_{k-2} is reached through π_5 where

$$\pi_5 = x_0 \xrightarrow{-2} x_{-2} \xrightarrow{+k} x_{k-2}$$

– x_{k+1} is reached through σ_{k+1}

$$\sigma_{k+1} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k} x_{k-1} \xrightarrow{+2} x_{k+1}$$

– x_{k+2} is reached through σ_{k+2}

$$\sigma_{k+2} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k} x_{k-1} \xrightarrow{+2} x_{k+1} \xrightarrow{+1} x_{k+2}$$

– x_{2k} is reached through σ_{2k}

$$\sigma_{2k} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k} x_{k-1} \xrightarrow{+k} x_{2k-1} \xrightarrow{+1} x_{2k}$$

– x_{-k+2}: Taking advantage of the fact that node $x_{-k} \in S_{area}$, node x_{-k+2} is reached through π_6 where

$$\pi_6 = x_0 \xrightarrow{-k} x_{-k} \xrightarrow{+2} x_{-k+2}$$

  - Nodes x_{-1} and x_{-k+1} are occupied by the SHs that were at nodes x_{-2} and x_{-k} when the original *black virus* was triggered.

- **Case 2**: In the case of finding the *black virus* in the fourth segment $n-k \leq |S_{area}| < n-2$, two *black viruses* are generated at x_1 and x_2 since the rest of the neighbouring nodes have been explored and guarded. Thus, $\mathcal{T}$=$\{x_3$, x_4 x_{-k+2}, $x_{k+2}\}$.

  - x_3 is reached using π_1 or π_2 as discussed in **Case1**.
  - x_4 is reached using π_3 or π_4 as discussed in **Case1**.
  - x_{k+2} is reached using σ_{k+2}.
  - x_{-k+2} is reached using π_6 as discussed in **Case1**.
  - x_{-k+1}, x_{-1} and x_{-k+1} are occupied by the SHs that were at nodes x_k, x_{-2} and x_{-k} when the original *black virus* was triggered.

- **Case 3**: In the case of finding the *black virus* at node v_{n-2}, one *black virus* is generated at x_1 since the other neighbouring nodes have been explored and guarded. Thus, $\mathcal{T}$=$\{x_2\}$

  - x_2 is reached using π_7 where

$$\pi_7 = x_0 \xrightarrow{+2} x_{+2}$$

  - Nodes $x_{-1}, x_{-k+1}, x_{k+1}$ and x_3 are occupied by the SHs that were at nodes x_2, x_k, x_{-2} and x_{-k} when the original *black virus* was triggered.

- **Case 4**: Here we have a special case where the *black virus* is located at node v_{n-1}. In this case, all neighbouring nodes are guarded and no more *black viruses* are created. No more moves are made in the second phase since all moves are made in the first phase.

- **Case5**: In the case of finding the *black virus* in the second segment $2 \leq |S_{area}| < k$, four *black viruses* are generated at $\mathcal{BV}$=$\{x_1, x_2, x_k$ and $x_{-k}\}$ since only one SH has been deployed so far at node x_{-2}. Thus, $\mathcal{T}$=$\{x_3,\ x_4,\ x_{k-2},\ x_{k-1},\ x_{k+1},\ x_{k+2},\ x_{2k},\ x_{-k+1},\ x_{-k+2},\ x_{-k-1},\ x_{-k-2},\ x_{-2k}\}$. To reach any of these targets, we should avoid any path that has x_{-k}.

  - Node x_3 is reached as follows:

$$\pi[x_0, x_3] = \min\{\sigma_3, \pi_2\}$$

    where

$$\sigma_3 = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k} x_{k-1} \xrightarrow{+2} x_{k+1} \xrightarrow{+2} x_{k+3} \xrightarrow{-k} x_3$$

  - Node x_4 is reached as follows:

$$\pi[x_0, x_4] = \min\{\sigma_4, \pi_4\}$$

    where

$$\sigma_4 = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k} x_{k-1} \xrightarrow{+2} x_{k+1} \xrightarrow{+2} x_{k+3} \xrightarrow{+1} x_{k+4} \xrightarrow{-k} x_4$$

  - Node x_{k-2} is reached using σ_{k-2}
  - Node x_{k-1} is reached using σ_{k-1}
  - Node x_{k+1} is reached using σ_{k+1}
  - Node x_{k+2} is reached using σ_{k+2}
  - Node x_{2k} is reached using σ_{2k}
  - Node x_{-k+1} is reached using σ_{-k+1} where

$$\sigma_{-k+1} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-k} x_{-k-1} \xrightarrow{+2} x_{-k+1}$$

– Node x_{-k+2} is reached using π_8 where

$$\pi_8 = x_0 \xrightarrow{-2} x_{-2} \xrightarrow{-k} x_{-k-2}$$

– Node x_{-k-1} is reached using σ_{-k-1} where

$$\sigma_{-k-1} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-k} x_{-k-1}$$

– Node x_{-k-2} is reached using σ_{-k-2} where

$$\sigma_{-k-2} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-k} x_{-k-1} \xrightarrow{-1} x_{-k-2}$$

– Node x_{-2k} is reached using σ_{-2k} where

$$\sigma_{-2k} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-k} x_{-k-1} \xrightarrow{-k} x_{-2k-1} \xrightarrow{+1} x_{-2k}$$

– Node x_{-1} is already guarded by a SH that was at node x_{-2} when the original *black virus* was triggered.

- **Case 6**. In the case of finding the *black virus* at node v_1, five *black viruses* are generated at $\mathcal{BV}$=$\{x_1, x_2, x_k, x_{-2}$ and $x_{-k}\}$ since no SHs have been deployed. Thus, $\mathcal{T}$=$\{x_{-1}$, x_3, x_4, x_{k-2}, x_{k-1}, x_{k+1}, x_{k+2}, x_{2k}, x_{-3}, x_{-4}, x_{-k+1}, x_{-k+2}, x_{-k-1}, x_{-k-2}, $x_{-2k}\}$. To reach any of these targets, we should avoid any path that has x_{-2} or x_{-k} as follows:

  – Node x_{-1} is reached using σ_{-1} where

$$\sigma_{-1} = x_0 \xrightarrow{-1} x_{-1}$$

  – Node x_3 is reached using σ_3 or π_2 as discussed in **Case 5**.

– Node x_4 is reached using σ_4 or π_4 as discussed in **Case 5**.

– Node x_{k-2} is reached using σ_{k-2}

– Node x_{k-1} is reached using σ_{k-1}

– Node x_{k+1} is reached using σ_{k+1}

– Node x_{k+2} is reached using σ_{k+2}

– Node x_{2k} is reached using σ_{2k}

– Node x_{-k+1} is reached using σ_{-k+1}

– Node x_{-k+2} is reached using σ_{-k+2} where

$$\sigma_{-k+2} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-k} x_{-k-1} \xrightarrow{+2} x_{-k+1} \xrightarrow{+1} x_{-k+2}$$

– Node x_{-k-1} is reached using σ_{-k-1}

– Node x_{-k-2} is reached using σ_{-k-2}

– Node x_{-2k} is reached using σ_{-2k}

– Node x_{-3} is reached using σ_{-3} where

$$\sigma_{-3} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-2} x_{-3}$$

– Node x_{-4} is reached using σ_{-4} where

$$\sigma_{-4} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-2} x_{-3} \xrightarrow{-1} x_{-4}$$

The following table summarizes the number of moves required to reach each possible target. Notice that we distinguish between two situations: when $|S_{area}| < 2$ and when $|S_{area}| \geq k$.

Destination	Number of moves	
	$\lvert S_{area}\rvert < 2$	$\lvert S_{area}\rvert \geq k$
x_0	1	1
x_3	≤ 5	≤ 4
x_4	≤ 6	≤ 4
x_{k-2}	3	2
x_{k-1}	2	2
x_{k+1}	3	3
x_{k+2}	4	4
x_{2k}	4	4
x_{-1}	1	1
x_{-3}	2	-
x_{-4}	3	-
x_{-k-1}	2	-
x_{-k-2}	3	-
x_{-k+1}	3	1
x_{-k+2}	4	2
x_{-2k}	4	-
x_1	1	1
x_2	1	1
x_k	1	1
x_{-2}	1	-
x_{-k}	1	-
Total	≤ 55	≤ 31

Table 5.2: Move complexity in shortly-chorded triple loops $C_n(1, 2, k)$.

Theorem 22. *In any triple loop $C_n(1, 2, k)$ where $|S_{area}| \geq k$, the number of moves required* *to surround and eliminate BVs is ≤ 31.*

Proof. In any triple loop chordal ring $C_n(1,2,k)$ where $|S_{area}| \geq k$, the maximum number of moves required to reach all targets is 31. One move is made by the LEA to reach x_0. Node x_3 is reached within four moves. Node x_4 is reached within four moves. Node x_{k-1} is reached within two moves. Node x_{k-2} is reached within two moves. Node x_{k+1} is reached within three moves. Node x_{k+2} is reached within four moves. Node x_{2k} is reached within four moves. Node x_{-k+2} is reached within two moves. Two moves are made by SHs to move from nodes x_{-2} and x_{-k} to x_{-1} and x_{-k+1} when the original *black virus* was triggered. After all agents arrive at their destinations, the LEA sends three CAs to the *black viruses* at x_1, x_2 and x_k in three moves. Notice that since we always compare the resulting routes π_z, where $z \in \mathbb{Z}$ to the σ_i, none of π_z would be greater than the corresponding σ_i. □

Theorem 23. *In any triple loop $C_n(1,2,k)$ where $|S_{area}| < 2$, the number of moves required to surround and eliminate BVs is ≤ 55.*

Proof. In any triple loop chordal ring $C_n(1,2,k)$ where $|S_{area}| < 2$, the maximum number of moves required to reach all targets is 55. One move is made by the LEA to reach x_0. Node x_3 is reached within five moves. Node x_4 is reached within six moves. Node x_{k-1} is reached within two moves. Node x_{k-2} is reached within three moves. Node x_{k+1} is reached within three moves. Node x_{k+2} is reached within four moves. Node x_{2k} is reached within four moves. Node x_{-k+2} is reached within four moves. Node x_{-1} is reached within one move. Node x_{-3} is reached within two moves. Node x_{-4} is reached within three moves. Node x_{-k-1} is reached within two moves. Node x_{-k-2} is reached within three moves. Node x_{-k+1} is reached within three moves. Node x_{-2k} is reached within four moves. After all agents arrive at their destinations, the LEA sends five CAs to the *black viruses* at x_1, x_2, x_k,x_{-2} and x_{-k} in five moves. Notice that since we always compare the resulting routes π_z, where $z \in \mathbb{Z}$ to the σ_i, none of π_z would be greater than the corresponding σ_i. □

5.2.2.2 On Optimality and Other Observations

In triple loops in general, the surrounding strategy provides optimal routes since it is mainly coordinated by the LEA. For the triple loop ring $C_n(1, 2, k)$, we have provided optimal routes when $k << n$. We ran a simulation to construct a partial *Breadth-First Search* tree rooted in x_0 in triple loops with "missing nodes" corresponding to the black viruses triggered in the various scenarios. The spanning tree was constructed until all targets appeared as leaves. In this type of spanning tree, any path from x_0 to a target leaf is the shortest path from x_0 to that destination. The scenarios tested include:
1) Shortly-chorded triple loop $C_n(1, 2, k)$ when $|S_{area}| < 2$.
2) Shortly-chorded triple loop $C_n(1, 2, k)$ when $|S_{area}| \geq k$.
In those scenarios we have verified that the paths indicated above correspond to the shortest paths.

If all agents have full knowledge of the topology and of the targets, the aforementioned approach can be transformed into a *local strategy* as mentioned in 4.2.1.3.

5.3 Triple Loops: the Case of $C_n(1, k-1, k)$

Let us now discuss the chordal ring $C_n(1, k-1, k)$ where $k << n$.

5.3.1 Exploring and Shadowing

The *Exploring and Shadowing* phase is the same in all triple loop rings. The only difference is the location of SHs which depends on the chords structure.

```
EXPLORING AND SHADOWING

let $HB = v_0$
Agents $EA$ and LEA at safe node $v_i$.

if ($1 \leq i+1 < k-1$)
    $N_{ex}(v_{i+1}) = \{v_i\}$
    no $SH$ is deployed
if ($i = k-2$)
    $N_{ex}(v_{i+1}) = \{v_i, v_0\}$
    1 $SH$ is deployed to protect $v_0$
if ($k \leq i+1 < n-k$)
    $N_{ex}(v_{i+1}) = \{v_i, v_{i+2-k}, v_{i+1-k}\}$
    2 $SH$s are deployed to protect $v_{i+2-k}, v_{i+1-k}$
if ($i+1 = n-k$)
    $N_{ex}(v_{i+1}) = \{v_i, v_{i+2-k}, v_{i+1-k}, v_{i+1+k}\}$
    3 $SH$s are deployed to protect $v_{i-k+2}, v_{i+1-k}, v_{i+1+k}$
if ($n-k+1 \leq i+1 < n-1$)
    $N_{ex}(v_{i+1}) = \{v_i, v_{i-k+2}, v_{i+1-k}, v_{i+1+k}, v_{i+k}\}$
    4 $SH$s are deployed to protect $v_{i-k+2}, v_{i+1-k}, v_{i+1+k}, v_{i+k}$
if ($i = n-2$)
    $N_{ex}(v_{i+1}) = \{v_i, v_{i-k+2}, v_{i+1-k}, v_{i+1+k}, v_{i+k}, v_0\}$
    5 $SH$s are deployed to protect $v_{i-k+2}, v_{i+1-k}, v_{i+1+k}, v_{i+k}, v_0$

EA moves to $v_{i+1}$.
```

The observations below were made following the employment of this strategy:

Theorem 24. *In the worst case scenario, the black virus is detected in* $5n - 9$ *moves.*

Proof. The worst case scenario for the number of moves required occurs when the *black virus* is found at node v_i, where $i = n - 1$. In this case, the BV triggers no new *black viruses* since all of the neighbouring nodes in the safe area have been explored and are protected by SHs. The complexity of this case would be $(3(n-1)-2)$ for the movement of LEA and EA, $(n-1-k)$ for one SH to guard node v_{n-1-k}, $(n-k)$ for the second SH to guard node v_{n-k}, $(k-1)$ for the third SH to guard node v_{k-1} and $(k-2)$ for the fourth SH to guard node v_{k-2} for a total of $5n - 9$ moves. □

Theorem 25. *In any triple loop chordal ring* $C_n = \{1, k-1, k\}$*, the worst case scenario in terms of the number of agents required occurs when five new black viruses are created after the original one is triggered.*

Proof. The worst case scenario for the number of cleaning agents (CAs) and surrounding agents (SAs) required occurs when the *black virus* is at node v_i where $1 \leq i < k - 1$. In this case, where the *black virus* is x_0, it triggers five new *black viruses* at x_1, x_{k-1}, x_k, x_{-k+1} and x_{-k} because no SH have been deployed. x_{-1} is always occupied by the LEA. In this case, the maximum number of *black viruses* is created. □

5.3.2 Surrounding and Eliminating

As previously discussed, the *Surrounding and Eliminating* phase remains the same as it was for the general case.

```
SURROUNDING AND ELIMINATING

LEA and $SH$s covering all $N_{ex}(v)$
$BV$ comes back from $v = x_0$.


if ($1 \leq |S_{area}| < k-1$) (*LEA is covering $x_{-1}$*)
    $N_{un}(x_0) = \{x_1, x_{k-1}, x_k, x_{-k+1}, x_{-k}\}$
if ($|S_{area}| = k-1$) (*LEA and $SH$ covering $x_{-1}, x_{-k+1}$*)
    $N_{un}(x_0) = \{x_1, x_{k-1}, x_k, x_{-k}\}$
if ($k \leq |S_{area}| < n-k$) (*LEA and 2 $SH$s covering $x_{-1}, x_{-k+1}, x_{-k}$*)
    $N_{un}(x_0) = \{x_1, x_{k-1}, x_k\}$
if ($|S_{area}| = n-k$) (*LEA and $SH$s covering $x_{-1}, x_{-k+1}, x_{-k}, x_k$*)
    $N_{un}(x_0) = \{x_1, x_{k-1}\}$
if ($n-k+1 \leq |S_{area}| < n-1$) (*LEA and $SH$s covering $x_{-1}, x_{-k+1}, x_{-k}, x_k, x_{k-1}$*)
    $N_{un}(x_0) = \{x_1\}$
Else (*LEA and $SH$s covering $x_{-1}, x_{-k+1}, x_{-k}, x_k, x_{k-1}, x_1$*)
    $N_{un}(x_0) = \emptyset$


All $SH$ make one move in the clockwise direction.
For each $u \in N_{un}(x_0)$:
    DEPLOY an agent to each $z \in \{N(u) \setminus N_{un}(x_0)\}$
When $N(u)$ is covered:
    DEPLOY one agent to $u$
```

The exploring team will find the *black virus* at (v_i) which is located in one of the five segments of the chordal ring:

- **Segment 1:** this segment contains nodes $v_1, v_2, ..., v_{k-2}$ and the size of the safe area is $1 \leq |S_{area}| \leq k-2$.

- **Segment 2:** this segment contains node v_{k-1} and the size of the safe area is $|S_{area}| = k-1$.

- **Segment 3:** this segment contains nodes $v_k, v_{k+1}, ..., v_{n-k-1}$ and the size of the safe area is $k \leq |S_{area}| \leq n-k-1$.

- **Segment 4:** this segment contains node v_{n-k} and the size of the safe area is $|S_{area}| = n-k$.

- **Segment 5:** this segment contains nodes $v_{n-k+1}, v_{n-k+2}, ..., v_{n-1}$ and the size of the safe area is $n-k+1 \leq |S_{area}| \leq n-1$.

The number of agents required to disinfect the triple loop chordal ring($C_n(1, k-1, k)$) is fixed whereas the number of moves varies depending on the deployment method.

Theorem 26. *Regardless of the deployment strategy, a maximum of 19 agents are employed in any triple loop $C_n(1, k-1, k)$ for black virus disinfection.*

Proof. The number of agents required is determined by the location of the original BV, regardless of the deployment method.

- If the BV is found in *Segment 1*, we would have the worst complexity in terms of CAs and SAs since activating the *black virus* will create five more *black viruses* at $x_1, x_{k-1}, x_k, x_{-k+1}$ and x_{-k}. In addition to the LEA at x_0, the LEA deploys 12 SAs to occupy x_{-1}, x_2, x_{k-2}, x_{k+1}, x_{2k-2}, x_{2k-1}, x_{2k}, x_{-k+2}, x_{-k-1} x_{-2k+2}, x_{-2k+1} and x_{-2k}. Therefore, $Spread(C) = 6$ and $Size(C) = 19$.

- If the BV is found in *Segment 2*, activating the original *black virus* would create four more *black viruses* at x_1, x_{k-1}, x_k and x_{-k} while x_{-k+1} is guarded by a SH. In addition to the SH at x_{-k+2} and the LEA at x_0, the LEA deploys 11 SAs to occupy x_{-1}, x_2, x_{k-2}, x_{k+1}, x_{2k-2}, x_{2k-1}, x_{2k}, x_{-k+1}, x_{-k-1}, x_{-2k+1} and x_{-2k}. Therefore, $Spread(C) = 5$ and $Size(C) = 18$.

- If the BV is found in *Segment 3*, activating the original *black virus* would create three more *black viruses* at x_1, x_{k-1} and x_k while x_{-k+1} and x_{-k} are guarded by SHs. Therefore, in addition to the SHs at x_{-k+2}, x_{-k+1} and LEA at x_0, the LEA deploys 7 SAs to occupy x_{-1}, x_2, x_{k-2}, x_{k+1}, x_{2k-2}, x_{2k-1} and x_{2k}. Therefore, $Spread(C) = 4$ and $Size(C) = 14$.

- *Segment 4* is part of the D_{area}. If the BV is found in this segment, activating the original *black virus* would create two more *black viruses* at x_1 and x_{k-1} while x_{-k+1}, x_{-k} and x_k are guarded by SHs. Therefore, in addition to the SHs at $x_{-k+2}, x_{-k+1}, x_{k+1}$ and the LEA at x_0, the LEA deploys 6 SAs to occupy x_{-1}, x_2, x_{k-2}, x_k, x_{2k-2} and x_{2k-1}. Therefore, $Spread(C) = 3$ and $Size(C) = 13$.

- *Segment 5* is also a part of the D_{area}. Here we have two possible cases:

  - When $n - k + 1 \leq i < n - 1$:

    If the BV is located at a node in this segment, activating the original *black virus* would create only one *black virus* at x_1, while x_{-k+1}, x_{-k}, x_k and x_{k-1} are guarded by SHs. In addition to the SHs at $x_{-k+2}, x_{-k+1}, x_{k+1}, x_k$, and LEA at x_0, the LEA deploys one SA to occupy x_2. Therefore, $Spread(C) = 2$ and $Size(C) = 8$.

  - When $i = n - 1$:

    In this case, no more *black viruses* are created since all of the neighbouring nodes are guarded by SHs and the LEA. No SAs need to be deployed and all the moves are made in the first phase. Therefore, $Spread(C) = 1$ and $Size(C) = 7$.

□

For the deployment part of this phase we suggest a non-local strategy where SAs follow specific paths set up by the LEA.

5.3.2.1 Move-Optimal Deployment

As with the general triple loop, for the surrounding phase we specify each path for each target. Let us now consider all the possible targets $\mathcal{T}$=\{x_{-1}, x_2, x_{k-2}, x_{k+1}, x_{2k-2}, x_{2k-1}, x_{2k}, x_{-k+2}, x_{-k-1} x_{-2k+2}, x_{-2k+1}, x_{-2k}\}. The special paths (σ_i) for this type of triple loop are identified below.

Target x_i	Special Path σ_i
x_{-1}	$x_0 \xrightarrow{-1} x_{-1}$
x_2	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k-1} x_{k-2} \xrightarrow{+k} x_{2k-2} \xrightarrow{+1} x_{2k-1} \xrightarrow{+1} x_{2k} \xrightarrow{-k+1} x_{k+1} \xrightarrow{-k+1} x_2$
x_{k-2}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k-1} x_{k-2}$
x_{k+1}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k-1} x_{k-2} \xrightarrow{+k} x_{2k-2} \xrightarrow{+1} x_{2k-1} \xrightarrow{+1} x_{2k} \xrightarrow{-k+1} x_{k+1}$
x_{2k-2}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k-1} x_{k-2} \xrightarrow{+k} x_{2k-2}$
x_{2k-1}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k-1} x_{k-2} \xrightarrow{+k} x_{2k-2} \xrightarrow{+1} x_{2k-1}$
x_{2k}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k-1} x_{k-2} \xrightarrow{+k} x_{2k-2} \xrightarrow{+1} x_{2k-1} \xrightarrow{+1} x_{2k}$
x_{-k-1}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-k} x_{-k-1}$
x_{-k+2}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-k} x_{-k-1} \xrightarrow{-k+1} x_{-2k} \xrightarrow{+1} x_{-2k+1} \xrightarrow{+1} x_{-2k+2} \xrightarrow{+k} x_{-k+2}$
x_{-2k+2}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-k} x_{-k-1} \xrightarrow{-k+1} x_{-2k} \xrightarrow{+1} x_{-2k+1} \xrightarrow{+1} x_{-2k+2}$
x_{-2k+1}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-k} x_{-k-1} \xrightarrow{-k+1} x_{-2k} \xrightarrow{+1} x_{-2k+1}$
x_{-2k}	$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-k} x_{-k-1} \xrightarrow{-k+1} x_{-2k}$

Depending on the chords structure and the size of the safe area, it may be possible to devise shorter paths in some cases. A detailed analysis of all possible situations is provided in the following section.

Let us now consider the different routes ($\pi[x_0, x_i]$) to each target in six different cases, depending on the location of the *black virus*.

- **Case 1**: In this case we examine the process of finding the *black virus* in the third segment of the chordal ring: $k \leq |S_{area}| < n - k$. In this case, triggering the original *black virus* creates three more *black viruses* : x_1, x_{k-1} and x_k, and thus $\mathcal{T}$={x_{-1}, x_2, x_{k-2}, x_{k+1}, x_{2k-2}, x_{2k-1}, x_{2k}, x_{-k+2}}.

- x_{-1} is reached through σ_{-1}

$$\sigma_{-1} = x_0 \xrightarrow{-1} x_{-1}$$

- x_2 can be reached as follows:

$$\pi[x_0, x_2] = \min\{\pi_1, \pi_2\}$$

  Taking advantage of the fact that node $x_{-k+1} \in S_{area}$, node x_2 is reached through π_1 where

$$\pi_1 = x_0 \xrightarrow{-k+1} x_{-k+1} \xrightarrow{+1} x_{-k+2} \xrightarrow{+k} x_2$$

$$\pi_2 = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{k-1} x_{k-2} \xrightarrow{-1} x_{k-3}, ..., \xrightarrow{-1} x_2$$

- x_{k-2} is reached through σ_{k-2}

$$\sigma_{k-2} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k-1} x_{k-2}$$

- x $_{k+1}$ is reached through π_3 where

$$\pi_3 = x_0 \xrightarrow{-k+1} x_{-k+1} \xrightarrow{+1} x_{-k+2} \xrightarrow{+k} x_2 \xrightarrow{+k} x_{k+2} \xrightarrow{-1} x_{k+1}$$

- x_{2k-2} is reached through σ_{2k-2}

$$\sigma_{2k-2} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k-1} x_{k-2} \xrightarrow{+k} x_{2k-2}$$

- x_{2k-1} is reached through σ_{2k-1}

$$\sigma_{2k-1} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k-1} x_{k-2} \xrightarrow{+k} x_{2k-2} \xrightarrow{+1} x_{2k-1}$$

- x_{2k} is reached through σ_{2k}:

$$\sigma_{2k} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k-1} x_{k-2} \xrightarrow{+k} x_{2k-2} \xrightarrow{+1} x_{2k-1} \xrightarrow{+1} x_{2k}$$

- x_{-k+2} and x_{-k+1} are occupied by the SHs that were at nodes x_{-k+1} and x_{-k} when the original *black virus* was triggered.

• **Case 2**: In the case of finding the *black virus* in the fourth segment where$|S_{area}| = n - k$, two *black viruses* are generated at x_1 and x_{k-1} since the other neighbouring nodes have been explored and guarded. Thus, $\mathcal{T}$=$\{x_{-1}$, x_2, x_{k-2}, x_{2k-2}, x_{2k-1}, $x_k\}$.

- x_{-1} is reached using σ_{-1}.
- x_2 is reached using π_1 or π_2 as discussed in **Case1**.
- x_{k-2} is reached using σ_{k-2}.
- x_k is reached using π_4 where

$$\pi_4 = x_0 \xrightarrow{k} x_k$$

- x_{2k-2} is reached using σ_{2k-2}.
- x_{2k-1} is reached using π_5 where

$$\pi_5 = x_0 \xrightarrow{k} x_k \xrightarrow{+k-1} x_{2k-1}$$

- x_{-k+1}, x_{-k+2} and x_{k+1} are occupied by the SHs that were at nodes x_{-k}, x_{-k+1} and x_k when the original *black virus* was triggered.

- **Case 3**: In the case of finding the *black virus* at node v_i where $n - k + 1 \leq |S_{area}| < n - 1$, one *black virus* is generated at x_1 since the other neighbouring nodes have been explored and guarded. Thus, $\mathcal{T}$=$\{x_2\}$.
  - x_2 is reached using $\pi_1$or π_2 as mentioned in **Case 1**.
  - Nodes $x_{-k+1}, x_{-k+2}, x_{k+1}$ and x_k are occupied by the SHs that were at nodes x_{-k}, x_{-k+1}, x_k and x_{k-1} when the original *black virus* was triggered.
- **Case 4**: Here we have a special case in which the *black virus* is located at node v_{n-1}. In this case, all neighbouring nodes are guarded and no more *black viruses* are created. No more moves are made in the second phase since all moves are made in the first phase.
- **Case 5**: In the case of finding the *black virus* in the second segment $|S_{area}| = k - 1$, four *black viruses* are generated at $\mathcal{BV}$=$\{x_1, x_{k-1}, x_k, x_{-k}\}$ since only one SH has been deployed so far at node x_{-k+1}. Thus, $\mathcal{T}$=$\{x_{-1}$, x_2, x_{k-2}, x_{k+1}, x_{2k-2}, x_{2k-1}, x_{2k}, x_{-k+1}, x_{-k-1}, x_{-2k+1}, $x_{-2k}\}$. To reach any of these targets, we should avoid any path that has x_{-k}.
  - Node x_{-1} is reached using σ_{-1}.
  - Node x_2 is reached using π_1 or $\pi_2$as discussed in **Case1**.
  - Node x_{k-2} is reached using σ_{k-2}.
  - Node x_{k+1} is reached using π_3.
  - Node x_{2k-2} is reached using σ_{2k-2}.
  - Node x_{2k-1} is reached using σ_{2k-1}.
  - Node x_{2k} is reached using σ_{2k}.
  - Node x_{-k-1} is reached using σ_{-k-1}.

- Node x_{-k+1} is reached using π_6 where

$$\pi_6 = x_0 \xrightarrow{-k+1} x_{-k+1}$$

- Node x_{-2k} is reached using σ_{-2k} where

$$\sigma_{-2k} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-k} x_{-k-1} \xrightarrow{-k} x_{-2k-1} \xrightarrow{+1} x_{-2k}$$

- Node x_{-2k+1} is reached using σ_{-2k+1} where

$$\sigma_{-2k+1} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-k} x_{-k-1} \xrightarrow{-k} x_{-2k-1} \xrightarrow{+1} x_{-2k} \xrightarrow{+1} x_{-2k+1}$$

- Node x_{-k+2} is already guarded by a SH that was at node x_{-k+1} when the original *black virus* was triggered.

- **Case 6**. Finding the *black virus* in the first segment, where $1 \leq |S_{area}| < k-1$, five *black viruses* are generated at $\mathcal{BV}$=$\{x_1, x_{k-1}, x_k, x_{-k+1}$ and $x_{-k}\}$ since no SHs have been deployed. Thus, $\mathcal{T}$=$\{x_{-1}$, x_2, x_{k-2}, x_{k+1}, x_{2k-2}, x_{2k-1}, x_{2k}, x_{-k+2}, x_{-k-1} x_{-2k+2}, x_{-2k+1}, $x_{-2k}\}$. To reach any of these targets, we should avoid any path that has x_{-k+1} or x_{-k} as the following:

  - Node x_{-1} is reached using σ_{-1}.
  - Node x_2 is reached using $\min\{\pi_2, \sigma_2\}$.
  - Node x_{k-2} is reached using σ_{k-2}.
  - Node x_{k+1} is reached using σ_{k+1}.
  - Node x_{2k-2} is reached using σ_{2k-2}.
  - Node x_{2k-1} is reached using σ_{2k-1}.
  - Node x_{2k} is reached using σ_{2k}.

– Node x_{-k+2} is reached as the following:

$$\pi[x_0, x_2] = \min\{\sigma_{-k+2}, \pi_7\}$$

where

$$\sigma_{-k+2} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-k} x_{-k-1} \xrightarrow{-k+1} x_{-2k} \xrightarrow{+1} x_{-2k+1} \xrightarrow{+1} x_{-2k+2} \xrightarrow{+k} x_{-k+2}$$

and

$$\pi_7 = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-1} x_{-2} \xrightarrow{-1}, ..., \xrightarrow{-1} x_{-k+2}$$

– Node x_{-2k+2} is reached using σ_{-2k+2} where

$$\sigma_{-2k+2} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-k} x_{-k-1} \xrightarrow{-k+1} x_{-2k} \xrightarrow{+1} x_{-2k+1} \xrightarrow{+1} x_{-2k+2}$$

– Node x_{-2k+1} is reached using σ_{-2k+1}.

– Node x_{-2k} is reached using σ_{-2k}.

The following table summarizes the number of moves required to reach each possible target. Notice that we distinguish between two possible situations: when $|S_{area}| < k - 1$ and when $|S_{area}| \geq k$.

Destination	Number of moves	
	$\lvert S_{area}\rvert < k-1$	$\lvert S_{area}\rvert \geq k$
x_0	1	1
x_2	≤ 7	3
x_{k-2}	2	2
x_{k+1}	6	5
x_{2k-2}	3	3
x_{2k-1}	4	≤ 4
x_{2k}	5	5
x_{-1}	1	1
x_{-k+2}	≤ 6	1
x_{-k-1}	2	-
x_{-2k+2}	5	-
x_{-2k+1}	4	
x_{-2k}	3	-
x_1	1	1
x_{k-1}	1	1
x_k	1	1
x_{-k+1}	1	-
x_{-k}	1	-
Total	≤ 54	≤ 28

Table 5.3: Move complexity in shortly-chorded triple loops $C_n(1, k-1, k)$.

Theorem 27. *In any triple loop $C_n(1, 2, k)$ where $|S_{area}| \geq k$, the number of moves required to surround and eliminate the BVs is ≤ 28.*

Proof. In any triple loop chordal ring $C_n(1, k-1, k)$ where $|S_{area}| \geq k$, the maximum number of moves required to reach all targets is 28. One move is made by the LEA to reach x_0. Node x_2 is reached within three moves. Node x_{k-2} is reached within two moves. Node x_{k+1} is reached within five moves. Node x_{2k-2} is reached within three moves. Node x_{2k-1} is reached within four moves. Node x_{2k} is reached within five moves. Node x_{-1} is reached within one move. Two moves are made by the SHs to move from nodes x_{-k+1} and x_{-k} to x_{-k+2} and x_{-k+1} when the original *black virus* is triggered. After all agents arrive at their destinations, the LEA sends three CAs to the *black viruses* at x_1, x_{k-1} and x_k in three moves. Since we always compare the resulting routes π_z, where $z \in \mathbb{Z}$ to the σ_i, none of π_z would be greater than the corresponding σ_i. □

Theorem 28. *In any triple loop $C_n(1, k-1, k)$ where $|S_{area}| < k-1$, the number of moves required to surround and eliminate theBVs is ≤ 54.*

Proof. In any triple loop chordal ring $C_n(1, k-1, k)$ where $|S_{area}| < k-1$, the maximum number of moves required to reach all targets is 54. One move is made by the LEA to reach x_0. Node x_2 is reached within three moves. Node x_{k-2} is reached within two moves. Node x_{k+1} is reached within five moves. Node x_{2k-2} is reached within three moves. Node x_{2k-1} is reached within four moves. Node x_{2k} is reached within five moves. Node x_{-1} is reached with one move. Node x_{-k+2} is reached within four moves. Node x_{-k-1} is reached with one move. Node x_{-2k+2} is reached within two moves. Node x_{-2k+1} is reached within three moves. Node x_{-2k} is reached within two moves. After all agents arrive at their destinations, the LEA sends five CAs to the *black viruses* at x_1, x_{k-1}, x_k,x_{-k+1} and x_{-k} in five moves. Since we always compare the resulting routes π_z, where $z \in \mathbb{Z}$ to the σ_i, none of π_z would be greater than the corresponding σ_i. □

5.3.2.2 On Optimality and Other Observations

For the triple loop ring $C_n(1, k-1, k)$, we have provided optimal routes when $k << n$. We ran a simulation to construct a partial *Breadth-First Search* tree rooted in x_0 in triple loops with "missing nodes" corresponding to the black viruses triggered in the various scenarios:

1) Shortly-chorded triple loop $C_n(1, k-1, k)$ when $|S_{area}| < k-1$, 2) Shortly-chorded triple loop $C_n(1, k-1, k)$ when $|S_{area}| \geq k$. In those scenarios we have verified that the paths above correspond to the shortest paths.

If all agents have full knowledge of the topology and of the targets, the aforementioned approach can be transformed into a *local strategy* as mentioned in 4.2.1.3.

5.4 Discussion and Comparisons

In this section we combine and analyze the results we have obtained in this chapter. We have investigated the two phases required to disinfect triple loop chordal rings from *black viruses*. Throughout this chapter we have discussed triple loop rings without considering the values of p and k. We found that it is difficult to consider all of the possible combinations of p and k in order to obtain an exact bound on the optimal paths, therefore, we have approached this class of triple loops by considering two extreme cases: $C_(1, 2, k)$ and $C_(1, k-1, k)$ where $k << n$. In the two extremes, we have have shown optimal complexities, while in the general case, we have only shown an upper bound.

In the following table, we combine the number of moves of the two phases for a triple loop chordal ring with arbitrary values of p and k, $C_n(1, p, k)$, for a specific class of triple loops where $p = 2$, $C_n(1, 2, k)$, and for another extreme class of triple loops where $p = k-1$, $C_n(1, k-1, k)$.

Triple loop	$\|S_{area}\| < p$		$\|S_{area}\| \geq k$	
	phase1	phase2	phase1	phase2
$C_n(1, p, k)$	$\leq 3p - 5$	78	$\leq 5n - 6k - p - 7$	44
$C_n(1, 2, k)$	1	55	$\leq 5n - 6k - 9$	31
$C_n(1, k-1, k)$	$\leq 3k - 8$	54	$\leq 5n - 7k - 6$	28

Table 5.4: Move complexity in triple loop rings in two different safe area sizes

From this table we can see that in the second phase, we have a gap between the two extreme case results and the general case since in the general case we did not consider the fact that some targets represent the same nodes, or some targets are *black viruses.* It is completely dependent on the chords structure. On the other hand, in the two extreme cases, we have a better understanding of the chords structure so we have been able to obtain optimal complexity.

5.5 Conclusion

In this chapter we investigated the problem of disinfecting a specific class of chordal rings, the *triple loop*, from black viruses using synchronous execution.

The first phase is common to all chordal rings and consists of *safe exploration* performed by the team of agents:LEA, EA and sometimes SHs. Once the black virus is found, it moves to the unexplored neighbouring nodes, EA disappears, and the second phase begins. The number of mobile agents and the number of *black viruses* created depends on the location of the original *black virus* in respect to the size of the safe area.

In the case of the second phase, we have demonstrated that, in contrast with the double loop, the greedy approach does not always perform the routing correctly since it is vulnerable to forming infinite loops in some triple loop chordal rings. We have proposed a non-local move-optimal surrounding strategy.

We first studied the general case of triple loop rings where p and k are arbitrary values, but we found that it is complicated to consider all possible combinations of p and k in order to obtain an exact bound on the optimal paths to reach the target nodes. Therefore, we studied two classes of triple loop rings in order to approach the problem: $C_n(1, 2, k)$ and $C_n(1, k-1, k)$.

As for the double loop, the non-local approach can be transformed into a local strategy where agents decide the next step without referring back to the LEA. The agents would have to re-compute the shortest path to their targets at each intermediate node. This strategy is still move-optimal and local, yet computational intensive. The synchronous strategy can be easily transformed into an asynchronous strategy, adding a constant number of moves.

Chapter 6

Black Virus Disinfection in Consecutive-Chords Rings

In this chapter, the black virus disinfection problem is considered in the *consecutive-chords* ring $C_n(1, 2, 3, .., k-1, k)$, with $k < \lfloor \frac{n}{2} \rfloor$, in a *synchronous* environment.

For the deployment phase, unlike the other classes of chordal rings, the local strategy that we propose, the *One-Direction Greedy* strategy, is also a move-optimal solution. We thus only concentrate on local deployment. As usual, we evaluate the complexity of the solution by considering the overall number of agents employed, the number of agent casualties and the number of moves required. The following table summarizes some of the results.

Consecutive-Chords(C)	$\|S_{area} \geq k$	$\|S_{area} < k$
Spread(C)	$\leq k+1$	$\leq 2k$
Size(C)	$\leq 3k+1$	$\leq 4k+2$
Move(C)	$\leq ((k+3)n - \frac{3}{2}(k^2+k) - 3$	$\leq n+6k-1$

6.1 Exploring and Shadowing

The exploring phase is the same as in all the other chordal rings. We instantiate it here for the particular case of the consecutive-chords chordal ring.

```
EXPLORING AND SHADOWING

Let $HB = v_0$.
Agents $EA$ and LEA at safe node $v_j$.

if $j + 1 < k$ (* $N_{ex}(v_{j+1}) = \{v_j, v_{j-1}, ..., v_0\}$ *)
    $(j - 1)SH$s are deployed to protect $N_{ex}(v_{j+1})$
Else if $k \leq j + 1 < n - k$ (* $N_{ex}(v_{j+1}) = \{v_j, v_{j-1}, v_{j-2}, ..., v_{j+1-k}\}$ *)
    $k - 1$ $SH$s are deployed to protect $N_{ex}(v_{j+1})$
Else if $n - k \leq j + 1 \leq n - 1$ (* $N_{ex}(v_{j+1}) = \{v_j, v_{j-1}, \ldots, v_{j+1-k}, v_{j+1+k}, v_{j+k}, \ldots, v_0\}$ *)
    let $j + 1 = n - i$, where $i \in \mathbb{Z}_{\geq 0}$
    $(2k - i)$ $SH$s are deployed to protect $N_{ex}(v_{j+1})$
EA moves to $v_{j+1}$.
```

The observations below were obtained following the application of this strategy:

Theorem 29. *In the worst case scenario, the black virus is detected in $(k + 2)n - 2k - 3$ moves.*

Proof. The worst case scenario for the number of moves required occurs when the *black virus* is found at node (v_{n-1}) after exploring $n - 1$ nodes. In this case, the BV triggers no new *black viruses* since all of the neighbouring nodes in the safe area have been explored and are protected by SHs. The complexity of this case is $3(n - 1) - 2$ for the movement of LEA and EA, $(\sum_{i=2}^{k} n-1-i)$ for the movement of $k-1$ SHs in the counter clockwise direction and $(\sum_{i=1}^{k-1} i)$ for the movement of SHs in the other direction. □

Theorem 30. *In any consecutive-chord ring, the worst case scenario in term of the number of agents required occurs when* $(2k-1)$ *new black viruses are created after triggering the original virus.*

Proof. The worst case scenario for the number of *black viruses* created upon activating the original *black virus* is when the original is found at node v_i where $\leq i = 1$. In this case, the *black virus* (x_0) triggers $(2k-1)$ new *black viruses*: x_1, x_2, x_3, ..., x_k, and x_{-2}, x_{-3}, ..., x_{-k} because no SH have been deployed. Note that x_{-1} is always occupied by LEA.

□

6.2 Surrounding and Eliminating

The deployment phase of the consecutive-chords chordal ring is different from all the other structures considered in this thesis because when the *black virus* is triggered, the chordal ring gets disconnected in the clockwise direction.

```
SURROUNDING AND ELIMINATING

LEA and $SH$s covering all $N_{ex}(v)$
$BV$ comes back from $v = x_0$.


if ($|S_{area}| = 1$) (* LEA is covering $x_{-1}$ *)
    $N_{un}(x_0) = \{x_1, x_2, ..., x_k, x_{-2}, x_{-3}, ..., x_{-k+1}, x_{-k}\}$
if ($1 < |S_{area}| < k$)
    let $|S_{area}| = k - i$, where $i \in \mathbb{Z}_{\geq 0}$
    (* LEA is covering $x_{-1}$ and $SH$s covering $N_{ex}(x_0)$*)
    $N_{un}(x_0) = \{x_1, x_2, ..., x_k, x_{-k}, x_{-k+1}, x_{-k+2}, ..., x_{-k+i-1}\}$
if ($k \leq |S_{area}| < n - k$) (* LEA and $SH$ covering $x_{-1}, x_{-2}, ..., x_{-k}$ *)
    $N_{un}(x_0) = \{x_1, x_2, ..., x_k\}$
if ($n - k \leq |S_{area}| < n - 1$)
    let $|S_{area}| = n - i$, where $i \in \mathbb{Z}_{\geq 0}$
    (* LEA and $SH$s covering $x_{-1}, x_{-2}, ..., x_{-k}, x_i, x_{i+1}, ..., x_k$ *)
    $N_{un}(x_0) = \{x_1, x_2, ..., x_{i-1}\}$
Else (* LEA and $SH$s covering $x_{-1}, x_{-2}..., x_{-k}, x_1, x_2, ..., x_k$ *)
    $N_{un}(x_0) = \emptyset$


All $SH$ make one move in clockwise direction.
For each $u \in N_{un}(x_0)$:
    DEPLOY an agent to each $z \in \{N(u) \setminus N_{un}(x_0)\}$
When $N(u)$ is covered:
    DEPLOY one agent to $u$
```

As with the previous chapters, we will consider different cases depending on the location of the *black virus* . More precisely, the number of agents required to disinfect the consecutive-chords ring C_n depends entirely on the chords structure and the location of black viruses in respect to the safe area. We first partition the chordal ring into four distinct segments using v_0 as the homebase as seen in figure 6.1.

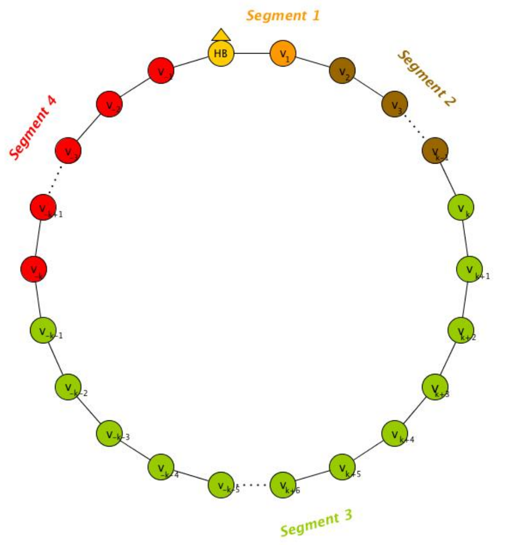

Figure 6.1: Dividing the consecutive-chords into four segments.

- *Segment 1* contains node v_1. If the *black virus* is in this segment, the size of the safe area is $|S_{area}| = 1$.
- *Segment 2* contains nodes $v_2, \dots v_{k-1}$. If the *black virus* is in this segment, the size of the safe area is $2 \leq |S_{area}| \leq k-1$.
- *Segment 3* contains nodes $v_k, \dots v_{n-k-1}$. If the *black virus* is in this segment, the size of the safe area is $k \leq |S_{area}| \leq n-k-1$.
- *Segment 4* contains nodes $v_{n-k}, \dots v_{n-1}$. If the *black virus* is in this segment, the size of the safe area is $n-k \leq |S_{area}| \leq n-1$.

Theorem 31. *For black virus disinfection in consecutive-chords ring $C_n(1, 2, ..., k)$, a maximum of $(4k+1)$ agents are employed.*

Proof. The number of agents is determined according to the location of the original BV and the chord structure.

- When the BV is located at any node in *Segment 1*, we have the worst complexity in terms of CA and SA since activating the *black virus* will create $(2k-1)$ *black viruses* at $x_1, x_2, ..., x_k, x_{-2}, x_{-3}, ..., x_{-k}$. Subsequently, the LEA moves to x_0 and then deploys $2k$ SAs while one SH is created in the x_{-1}, which is the homebase. The SAs are then to occupy $x_{k+1}, x_{k+2}, x_{k+3}, ..., x_{2k}, x_{-k-1}, x_{-k-2}, ..., x_{-2k}$. The total number of agents employed is then $(4k+2)$: $(2k)$ CAs, $(2k)$ SAs, one SH and one LEA. Therefore, $Spread(C) = 2k$ and $Size(C) = 4k+2$.

- If the BV is located at any node in *Segment 2* , activating the original *black virus* would create $k+i$ *black viruses* at x_1, x_2,..., x_k, x_{-k}, x_{-k+1} ,..., x_{-k+i}, where $i = k - |S_{area}|$. The same number of SHs as nodes in the safe area are then deployed and each makes one move through $+1$ chord when it receives a BV. LEA also moves to x_0, and then deploys $2k$ SAs to occupy x_{k+1}, x_{k+2}, x_{k+3}, ..., x_{2k}, x_{-k-1}, x_{-k-2}, ..., x_{-2k}. Therefore, the team of agents consists of $(4k+2)$ agents: $(k+i+1)$ CAs, $(2k)$ SAs, $(|S_{area}|)$ SHs and LEA. In this case, $Spread(C) = k+i+1 = 2k - |S_{area}| + 1$ and $Size(C) = 4k+2$.

- If the BV is located at any node in *Segment 3*, activating the original *black virus* would create k *black viruses* at x_1, x_2,..., x_k, while the other neighbouring nodes are already guarded. Once the LEA and SHs receive BVs, they make one move through their $+1$ chords. Then LEA deploys k SAs to occupy $x_{k+1}, x_{k+2}, x_{k+3}, ..., x_{2k}$. Therefore, the team of agents consists of $(3k+1)$ agents: $(k+1)$ CAs, (k) SAs, $(k-1)$ SHs and LEA. In this case, $Spread(C) = k+1$ and $Size(C) = 3k+1$.

- If the BV is located at any node in *Segment 4* , also called the *Danger area* (D_{area}), we have two possible cases:

– When $n - k \leq |S_{area}| < n - 1$: Let us assume that $|S_{area}| = n - i$. If the BV is located at any node in this segment, activating the original *black virus* would create $i - 1$ *black viruses* at x_1, x_2,..., x_{i-1}, while the other neighbouring nodes are guarded by SHs. In addition to the SHs at x_{-1}, x_{-2},..., x_{-k}, x_i, x_{i+1}, x_{i+2},..., x_k, and the LEA at x_0, the LEA deploys $i - 2$ SA to guard the unoccupied neighbours by SHs . Therefore, the team of agents consists of $(2k + i - 1)$ agents: (i) CAs, $(i - 2)$ SA, $(2k - i)$ SH and LEA. In this case, $Spread(C) = i = n - |S_{area}|$ and $Size(C) = 2k + i - 1 = 3k - |S_{area}| - 1$.

– When $|S_{area}| = n - 1$: No more *black viruses* are created since all of the neighbouring nodes of the *black virus* are guarded by SHs and the LEA. No SAs to be deployed and all of the moves are made in the first phase. In this case, the team consists of $2k+1$ agents: 1 CA, $2k-1$ SHs and LEA. Therefore,$Spread(C) = 1$ and $Size(C) = 2k + 1$.

□

As previously mentioned, triggering the *black virus* disconnects the chordal ring in one direction and in order to reach the targets to surround the new black viruses, the agents must move in a counter-clockwise direction.

The idea is for the surrounding agents to take the longest chord $-k$ to reach the area to be occupied as quickly as possible, and then reach their own target in one additional step through a shorter chord. Each agent can locally decide the following step at each intermediate node, simply on the basis of its target.

The one-direction greedy consists of two stages: the long jumps and the simple greedy. The long jumps portion forces SAs to move from x_0 to their targets through the edge$(-k)$ in a counter-clockwise direction making $(\lceil \frac{n-k}{k} \rceil - 2)$ hops. At that point an extra hop will cause them to reach their destination. Figure 6.2 shows an example of this strategy.

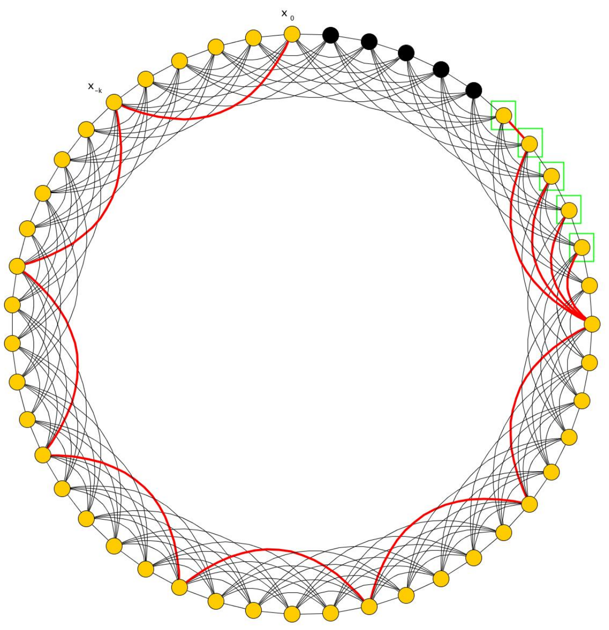

Figure 6.2: An example of One-direction strategy where red links show the paths to the targets.

```
ONE-DIRECTION GREEDY

Agent $A$ at node $x_{-i}$ $(i \geq 0)$ with target $x_j$
If $i = 0$
    If $|BV| \leq k$
        Agent $A$ moves through $\lambda = -k$
    If $|BV| > k$
        Agent $A$ moves through $\lambda = -1$, then moves through $\lambda = -k$
If $i < k(\lceil \frac{n-k}{k} \rceil - 2)$
    move through $\lambda = -k$
Else
    Let $FD = \{N(v) - \mathcal{BV} - y\}$ be the set of feasible destinations.
    Agent $A$ moves to $w \in FD$ that minimizes $dist(w, x_j)$
```

Theorem 32. *In any consecutive-chords ring $C_n(1, 2, 3, ..., k)$, with $k < \lfloor \frac{n}{2} \rfloor$, using the one-direction greedy approach in the* Surrounding and Eliminating *phase disinfects the whole* *topology from black viruses.*

Proof. After finding the original *black virus*, and depending on the size of the safe area, we get BVs in the clockwise direction or in both directions. In the first scenario we have $|S_{area}| \geq k$ and all BVs reside in one direction (clockwise). From the location of the LEA, the other direction is completely safe and clean, and all agents can traverse that part of the chordal ring until they reach their targets. In order to do so, the agents must be aware of the locations of BVs according to our algorithm. In the second scenario we have $|S_{area}| < k$, kBVs reside in the clockwise direction, and $\leq k$ BVs reside in the counter clockwise direction. From the location of the LEA the counter clockwise direction is not quite as safe as it was in the first scenario. Agents can still traverse that direction safely due to the fact that the node x_{-1} is always safe no matter the size of the safe area. Through node x_{-1}, the LEA will send agents to their targets, and the agents will avoid all the faulty nodes in that direction since the longest chord connected to x_{-1} surpasses the furthest possible BV (i.e., $x_{-1-k} < x_{-k}$).

□

In the following section we will consider the number of moves required in the two main cases: $|S_{area}| \geq k$ and $|S_{area}| < k$.

Theorem 33. *In any consecutive-chords ring $C_n(1, 2, 3, ..., k)$, with $k < \lfloor \frac{n}{2} \rfloor$, when $|S_{area}| \geq k$, the number of moves required to surround and eliminate the black viruses is $\leq k(\lceil \frac{n-k}{k} \rceil) + 2k$, regardless of the number of nodes.*

Proof. The number of moves required to decontaminate kBVs can be calculated as follows. One move is made by LEA to occupy x_0. The movements of SAs are calculated as follows:

1. Calculating the number of jumps. Agents need to traverse the ring starting from x_0

and using the longest chord as they move greedily:

$$\lfloor\frac{n-k}{k}\rfloor \tag{6.1}$$

2. Finding the number of $leftover$ nodes, which are not reached from the last jump, and require an extra jump:

$$leftover = ((n-k) \mod k) - 1 \tag{6.2}$$

   This equation excludes BVs and x_0.

3. Based on the aforementioned equations, we have the following information:

   - If 6.2≥ 0, then $((n-k) \mod k) - 1$ neighbours are reached within $\lfloor\frac{n-k}{k}\rfloor + 1$, and $k - (((n-k) \mod k) - 1)$ neighbours are reached within $\lfloor\frac{n-k}{k}\rfloor$. In total we have:
     $(k - leftover)\lfloor\frac{n-k}{k}\rfloor + leftover(\lfloor\frac{n-k}{k}\rfloor + 1)$
   - If 6.2< 0, all neighbours except one are reached within $\lfloor\frac{n-k}{k}\rfloor$. The other neighbour is reached in $\lfloor\frac{n-k}{k}\rfloor - 1$. In total, we have:
     $(k-1)\lfloor\frac{n-k}{k}\rfloor + (\lfloor\frac{n-k}{k}\rfloor - 1)$

In the theorem we substitute the floor function with the ceiling function in order to give a general upper bound for all of the possible values.

The SHs movements equal $k-1$ at most since all SHs make one move in the clockwise direction when the original *black virus* is activated. LEA sends k CAs in k moves to trigger the BVs.

□

Theorem 34. *In any consecutive-chords ring $C_n(1, 2, 3, ..., k)$, with $k < \lfloor\frac{n}{2}\rfloor$, when $|S_{area}| <$*

k, the number of moves required to surround and eliminate the black viruses is $\leq k(\lceil \frac{n-k-1}{k} \rceil)+$ $6k-1$, regardless of the number of nodes.

Proof. The number of moves required to decontaminate the worst case scenario in terms of *spread* and *size* (i.e., when $|S_{area}| = 1$) can be calculated as follows. One move is made by LEA to occupy x_0. In the worst case scenario we have $2k-1$ BVs, so in addition to $\{x_{k+1}, x_{k+2}, x_{k+3}, ..., x_{2k}\}$, there are more nodes to be guarded: $\{x_{-k-1}, x_{-k-2}, x_{-k-3}, ..., x_{-2k}\}$ The only way to reach those targets is through x_{-1}. k agents reach the consecutive neighbours $\{x_{k+1}, x_{k+2}, x_{k+3}, ..., x_{2k}\}$ by wrapping around in the counter clockwise direction after moving from x_0 to x_{-1} and using the one-direction greedy strategy. The movements of this group of SAs are calculated in the same way as the previous case when $|S_{area}| \geq k$ except that the long jumps start from x_{-1}.

1. Calculating the number of jumps. Agents need to traverse the ring starting from x_{-1} and using the longest chord as they move greedily:

$$\lfloor \frac{n-k-1}{k} \rfloor \tag{6.3}$$

2. Finding the number of $leftover$ nodes, which are not reached from the last jump, and require an extra jump:

$$leftover = ((n-k-1) \mod k) - 1 \tag{6.4}$$

   This equation excludes BVs and x_0.

3. According to the aforementioned equations, we know that:

   - If 6.4$\geq$ 0, then $leftover$ neighbours are reached within $\lfloor \frac{n-k-1}{k} \rfloor + 1$, and $(k - leftover)$ neighbours are reached within $\lfloor \frac{n-k-1}{k} \rfloor$. In total, we have:

$$(k - leftover)\lfloor\frac{n-k-1}{k}\rfloor + leftover(\lfloor\frac{n-k-1}{k}\rfloor + 1) + k$$

- If 6.2$<$ 0, all neighbours except one are reached within $\lfloor\frac{n-k-1}{k}\rfloor$. The other neighbour is reached in $\lfloor\frac{n-k-1}{k}\rfloor - 1$. In total, we have:
  $(k-1)\lfloor\frac{n-k-1}{k}\rfloor + (\lfloor\frac{n-k-1}{k}\rfloor - 1) + k$

Notice that k agents make an extra move from x_0 to x_{-1}.

In the theorem we substitute the floor function with the ceiling function in order to give a general upper bound for all of the possible values.

The consecutive nodes $\{x_{-k-1}, x_{-k-2}, x_{-k-3}, ..., x_{-2k}\}$ are reached in three moves each while node x_{-k-1} which is reached in two moves:

$$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-k} x_{-k-1} \xrightarrow{-i} ...$$

where $i = \{1, 2, 3, ..., k-1\}$ for a total of $3k-1$. LEA sends $2k-1$ CAs in $2k-1$ moves to trigger the BVs. □

6.3 Conclusion

In this chapter, we have addressed the problem of disinfecting *consecutive-chords* rings from the black virus. We have demonstrated that when we have an undirected consecutive-chords ring and a synchronous execution, the *Exploring and Shadowing* phase is the same as the one employed in other topologies, having an exploring team that uses the safe exploration technique starting from the homebase and moving in a counter-clockwise direction followed by shadows until the *black virus* is located and triggered. We divided the outer ring into segments to demonstrate the monotone nature of our strategy. The *Surrounding and Eliminating* phase begins once the original *black virus* has been located. As previously mentioned, this class of chordal rings is unlike the other classes in terms of routing and directing the surrounding agents. The local strategy proposed, the *one-Direction Greedy*, is also a move-optimal solution. In this approach, agents must be aware of their targets. Surrounding agents traverse in a counter-clockwise direction using the edge labels to calculate their next move until they reach their targets. We will now summarize and combine the results obtained in the previous two sections for the number of required moves.

BV **at node** v_i	**Phase 1**	**Phase 2**	**Total**
$i = 1$	1	$\leq k(\lceil \frac{n-k-1}{k} \rceil) + 6k - 1$	$\leq n + 6k - 1$
$1 < i < k$	$\leq \frac{k^2+k-2}{2} - 1$	$\leq k(\lceil \frac{n-k-1}{k} \rceil) + 5k - 1$	$\leq n + \frac{k^2+11k}{2} - 4$
$k \leq i < n - k$	$\leq (k+2)n - (\frac{3k^2+7k}{2}) - 3$	$\leq k(\lceil \frac{n-k}{k} \rceil) + 2k$	$\leq ((k+3)n - \frac{3}{2}(k^2 + k) - 3$
$n - k \leq i < n - 1$	$\leq (k+2)n - 4k - 4)$	$\leq (j-2)(\lceil \frac{n-k}{k} \rceil) + 2k$ where $j = n - \lvert S_{area} \rvert$	$\leq (\frac{k^2+2k+j-2}{k})n - 2k - 4)$
$i = n - 1$	$(k+2)n - 2k - 3$	1	$\leq (k+2)n - 2k - 2$

Table 6.1: Move complexity of disinfecting a consecutive-chords according to location of BV.

The above table shows the move complexity when we have synchronized mobile agents. The same approach can be applied using an asynchronous execution but includes an extra termination phase that can be applied by a coordinator (e.g. LEA). This extra phase ensures that that all SAs reach their targets before triggering the BVs.

Chapter 7

General Chordal Rings

Now that we have investigated the problem of disinfecting a network from a *black virus* in special classes of chordal rings, we will discuss the problem in general chordal rings, regardless of chords structure.

The solution we propose is based on the idea described for the general chordal ring in Chapter 3, and has already been explained throughout this thesis. It involves an exploring phase, followed by a surrounding phase. Our protocol is based on a general surrounding method that can be applied to any chordal ring structure, regardless of the number or distance of chords. It must be noted that the move-cost for this method is not optimal.

7.1 Exploring and Shadowing

The phase is described in detail in Chapter 3. The following are our observations in general chordal ring structures:

Theorem 35. *In any chordal ring $C_n = \{1, d_2, d_3, ..., d_m\}$, in the worst case scenario, the black virus is detected in $((2+m)n - 2m - 3)$ moves.*

Proof. The worst case scenario regarding the number of moves required occurs when the *black virus* is located at node (v_{n-1}) after exploring $n-1$ nodes. The complexity of this case would be $3n-5$ for the movement of LEA and EA, $(\sum_{i=2}^{m} n-1-d_i)$ for the movement of SHs to counter-clockwise neighbours and $(\sum_{i=2}^{m} d_i - 1)$ for SHs to clockwise neighbours.

In this case, the *black virus* triggers no new *black viruses* since all of the neighbouring nodes are occupied by SHs, $(2m-1)$ SHs. This case is considered the worst case scenario in terms of calculating the number of moves, but the best case in terms of cleaning and surrounding agents. $\square$

Theorem 36. *In any chordal ring* $C_n = \{1, d_2, d_3, ..., d_m\}$*, the worst case scenario regarding the number of agents required for disinfection would occur when* $2m-1$ *new black viruses are created after triggering the original virus.*

Proof. If the *black virus* is found at node (v_i) where $1 \leq v_i < d_2$, the spread of *black viruses* would be maximized since no SHs have been deployed and the explored neighbours $|N_{ex}(v_i)| = 1$, which is v_{i-1} and is occupied by the LEA. Therefore , the number of unexplored neighbours, o *black viruses* , is $2m-1$. This number of *black viruses* requires a high number of surrounding agents. $\square$

As usual, this phase comes to an end when the *black virus* is detected and triggered. At this point, new *black viruses* have been created and moved to the unexplored neighbouring nodes. It is at this point that the second phase begins.

7.2 Surrounding and Eliminating

As described in Chapter 3, once the *black virus* node is detected, the LEA moves to its location and the *Surrounding and Eliminating* phase begins.

Throughout our study of different classes of chordal rings, we have proposed two routing variations: local and non-local strategies. We have seen in previous chapters that these strategies work for some topologies and not others. For example, the local greedy approach causes correct routing in double loop chordal rings and infinite loops in triple loop and consecutive-chords rings. We can thus deduce that the simple greedy would not work in general chordal rings with arbitrary chord structures (see 5.1.2.1). The non-local move-optimal strategy would always work for any chordal rings since this approach requires precise information about the chord structure which is available to the LEA and complex to calculate. We have observed the following from both strategies:

- If node x_0 represents the location of the original *black virus* , node x_{-1} is always safe because it is occupied by an agent. This means that it can be used as a starting point to reach any target.
- After triggering the original *black virus*, the new *black viruses* divide the outer ring into enclosed and open segments. Enclosed segments are areas that include the following set of nodes: $\{x_{\pm d_i - 1}, x_{\pm d_i - 2}, x_{\pm d_i - 3}, ..., x_{\pm d_{i \pm 1} + 1}\}$. Open segments are areas that include the following set of nodes: $\{x_{\pm d_m \pm 1}, x_{\pm d_m \pm 2} ..., x_{\pm d_m}\}$.
- Once the agents reach node $x_{\pm d_i - 1}$, they are able to reach close targets greedily. In order to avoid getting stuck in infinite loops, agents can reach their close targets by using the one-direction greedy approach.

Making use of the observations above, we propose an efficient general strategy that is capable of disinfecting any chordal ring and gives upper bounds for the optimal paths. In this approach, all targets are reached through node x_{-1}. The set of targets is calculated based on the location of the original BV and the agents are scattered between BVs. The topology is thus divided into *enclosed areas* and *open areas*.

Note that with respect to node x_{-1}, some targets are in the clockwise direction while others are in the counter-clockwise direction.

In order to reach the targets, the LEA deploys agents through x_{-1}. The agents then move to the closest neighbour that is greater than or equal to its destination. Once the agent reaches $x_{\pm d_i - 1}$, it arrives to the area in which its target resides. The agent then moves greedily in one direction until it reaches its destination. Since this is a one-direction greedy approach, the agent only moves to neighbouring nodes that are greater or equal to its target. Only in one case an agent moves to a neighbour that is smaller than its target, when $t = x_{2d_m}$. Figure 7.1 shows an example of this strategy.

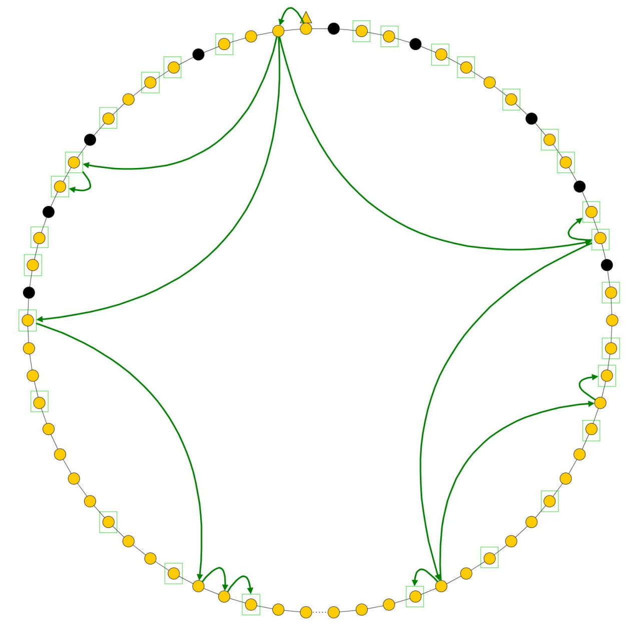

Figure 7.1: The paths to reach some targets using the general strategy.

General Deployment Strategy

```
Deploying agent $A$ arriving at $x_j$ from $y$ with destination $t$
If $t = x_z$ where $z \in \mathbb{Z}^+$
    if $j = 0$
        move to $x_{-1}$
    if $j = -1$
        move to $x_{d_i-1}$ such that $x_{d_i-1} \geq t$ , and $x_{d_i-1}$ minimizes $dist(x_{-1}, t)$
    Else
        if $t > x_{d_m}$
            move to $x_{d_m-1}$ then to $x_{2d_m-1}$ (i.e., move to the open area)
            If $t = x_{2d_m}$
                move to $x_{2d_m}$
            Else
            compute $next$
                let $OA = \{x_{2d_m-1}, x_{2d_m-2}, ..., x_{d_m+1}\}$the open area set
                Let $FD = \{N(x_j) - y - BV\}$ be the set of feasible destinations.
                let $C = FD \cap OA$
                Agent $A$ moves to $next \in C$ that minimizes $dist(x_j, t)$ and $next \geq t$
        Else
            compute $next$
                let $EA = \{x_{d_i-2}, x_{d_i-2}, x_{d_i-3}, x_{d_i-4}, ..., x_{d_{i-1}+1}\}$the enclosed area set
                Let $FD = \{N(x_j) - y - BV\}$
                let $C = FD \cap EA$
                Agent $A$ moves to $next \in C$ that minimizes $dist(x_j, t)$ and $next \geq t$
```

```
GENERAL DEPLOYMENT STRATEGY

Else (i.e., $t = x_z$ where $z \in \mathbb{Z}^-$)
    if $j = 0$
        move to $x_{-1}$
    if $j = -1$
        move to $x_{-d_i-1}$ such that $x_{-d_i-1} \geq t$ , and $x_{-d_i-1}$ minimizes $dist(x_{-1}, t)$
    Else
        if $t < x_{-d_m}$
            move to $x_{-d_m-1}$ (i.e., move to the open area)
            compute $next$
                let $OA = \{x_{-d_m-1}, x_{-d_m-2}, x_{-d_m-3}, ..., x_{-2d_m}\}$the open area set
                Let $FD = \{N(x_j) - y - BV\}$
                let $C = FD \cap OA$
                Agent $A$ moves to $next \in C$ that minimizes $dist(x_j, t)$ and $next \geq t$
        Else
            compute $next$
                let $EA = \{x_{-d_i-1}, x_{-d_i-2}, x_{-d_i-3}, x_{-d_i-4}, ..., x_{-d_{i+1}+1}\}$the enclosed area set
                Let $FD = \{N(x_j) - y - BV\}$
                let $C = FD \cap EA$
                Agent $A$ moves to $next \in C$ that minimizes $dist(x_j, t)$ and $next \geq t$
```

The following observations were made after using the general strategy:

Theorem 37. *In any chordal ring $C_(1, d_2, d_3, ...d_m)$ with $d_m << n$, the worst case scenario in terms of number of agents required occurs when triggering the original black virus creates $2m-1$ more black viruses. In this case, $size(C) \leq \lfloor \frac{4m^2-8m+3}{2} \rfloor + 6m - 1$ and $Spread(C) = 2m$.*

Proof. In any chordal ring C, the worst case scenario occurs when the original *black virus* is found before deploying any SH. In orther words, when $|S_{area}| < d_2$. In this case, the maximum number of *black viruses* would be created: $2m-1$ and $\mathcal{BV} = \{x_1,\ x_{\pm d_2},\ x_{\pm d_3},\ ...,\ x_{\pm d_m}\}$. If we assume that the chords are well separated (i.e., $d_i - d_{i-1} \geq 1$), we have $2(m-1)$ enclosed areas and 2 open areas. Each $bv \in BV$ has at most $2m-1$ neighbours which represent our targets. Some of these are common neighbours or other *black viruses* , depending on the structure of the chords. Because we are interested in calculating the $size(C)$, we must first find the number of targets. Each $bv \in BV$ has common neighbours and non-common neighbours. $N_{nc}(bv)$ and $N_c(bv)$ denote the set of non-common neighbours and the set of common neighbours of any $bv \in BV$. Note that for any i and j, $x_{d_i} + x_{-d_j}$ (a neighbour of node x_{d_i}) is the same as $x_{-d_j} + x_{d_i}$ (a neighbour of node x_{-d_j}). We have thus found that $|N_{nc}(bv)| \leq 2$ and $|N_c(bv)| \leq 2m-3$ for any $bv \in BV$. In other words, each *black virus* has a maximum of two non-common neighbours. Therefore, we can calculate the maximum possible number of targets as: $|\mathcal{T}| \leq \lfloor \frac{4m^2-8m+3}{2} \rfloor + 4m - 2$. $|\mathcal{T}|$ represents the number of SAs we need to surround the *black viruses* in the system.

The team of agents consists of $\leq \lfloor \frac{4m^2-8m+3}{2} \rfloor + 4m - 2$ SAs, one LEA and $(2m)$ CAs. Therefore $size(C) \leq \lfloor \frac{4m^2-8m+3}{2} \rfloor + 6m - 1$ and $Spread(C) = 2m$.

□

Theorem 38. *In any chordal ring* $C_(1, d_2, d_3, ...d_m)$ *where* $|S_{area}| < x_{d_2}$*, the number of moves required to surround and eliminate* BVs *is* $O(m^2)$*.*

Proof. Calculating the number of moves is not trivial in the general case. This approach is not optimal but it is efficient and gets the routing done correctly. We have an upper bound for the total number of moves required and it is constant. In the analysis of our strategy we find:

- $2m-1$ targets are reached in 2 moves, which are $N(x_{-1})$:

$$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{\pm d_i} x_{\pm d_i - 1}$$

- Two targets are reached in 4 moves each which are x_{2d_m} and x_{-2d_m}.

$$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{\pm d_m} x_{\pm d_m - 1} \xrightarrow{\pm d_m} x_{\pm 2d_m - 1} \xrightarrow{\pm 1} x_{\pm 2d_m}$$

- The rest of targets are nested in the enclosed and open areas and their locations are solely dependent on the chords structure. In order to reach them, the agents first move to node x_{-1} and then to one of the neighbours that satisfies the strategy's conditions. Once an agent is located at the beginning of an interval in which its destination resides, the one-direction greedy strategy begins.

  The destination between $x_{\pm d_i}$ and t can be minimized if there is a chord $d_i > 1$. In our thesis we will consider the worst case scenario in which an agent moves to its target through ± 1 chords. Therefore, each of the other targets, $\leq \lfloor \frac{4m^2-8m+3}{2} \rfloor + 2m - 3$, are reached in $O(m^2)$ moves.

□

7.3 Conclusion

In this chapter we addressed the *black virus disinfection* problem in any general chordal rings, regardless of the chords structure. In this chapter we have shown that the first phase remains the same in all chordal rings and that the second phase can be done non-locally. In this case, the LEA finds the shortest paths to the targets and sends SAs through them. In order to calculate the complexity of the optimal paths, we require information about the chords structure that is not available. As a result, we propose an efficient protocol that gives us upper bounds to the optimal length. In this protocol we considered some observations from all of the strategies discussed in previous chapters. This protocol is based on the fact that the presence of *black viruses* in the system divides the topology into *enclosed* and *open* areas and that all of those areas are reached through node x_{-1}. Once a SA reaches the area in which its target resides, it moves greedily in one direction until it reaches its destination. We have found that the total complexity of our general algorithm in term of moves is $O(mn)$ for the first phase, where m is the total number of chords in one direction, and $O(m^2)$ for the second phase.

Chapter 8

Conclusion

In this thesis, we investigated the black virus disinfection problem in undirected chordal rings, presenting a solution that is based on the use of the mobile agents model. We addressed the problem with the existence of a single black virus at unknown location of the chordal ring that, when triggered, generates and sends new viruses to unprotected neighbours. Although we assumed synchronous execution by the agents, this strategy can be easily extended to work in asynchronous settings.

The proposed solution is a distributed algorithm that consists of two phases: *Exploring and Shadowing* and *Surrounding and Eliminating*. Regarding cost, the efficiency measures considered include: the total number of black viruses originated in the system, the total number of mobile agents employed for disinfection and the total number of moves required by the agents. We found that the number of black viruses originated and agents required for disinfection is influenced by the location of the original black virus and that these numbers remain constant, regardless of the deployment method used in the surrounding phase. The number of moves varies according to the deployment strategy.

In our study we demonstrated that the first phase remains the same for all chordal rings and consists of the exploration of the graph using the safe exploration technique until the black virus is found. In order to achieve monotonicity, shadow agents are deployed during the search in order to guard the nodes that have been explored. The only difference between the types of chordal rings in the first phase is the number of shadow agents created at the beginning of the protocol. On the other hand, the second phase varies depending on the chords structure. Since black viruses are only destroyed when they arrive at a node that is being guarded, the agents must occupy all of the neighbouring nodes of the black viruses in the system. Routing is thus a critical part of this phase. Assuming that the leading agent has full topological knowledge, the routing can be done globally by the leading agent. The leading agent always finds the shortest path to all targets and sends the surrounding agents through them. This routing method is optimal, yet it requires that the surrounding agents have larger memories. We also proposed other local methodologies that do not require that the entire path be calculated by the leading agent. We analyze the complexity of all the deployment strategies, providing upper bounds on the path lengths and optimality when possible.

In chapter 4, we addressed the black virus disinfection problem in double loop chordal rings. In order to disinfect the entire topology we require a maximum of 12 agents. The maximum number of black viruses is four. For the surrounding phase we presented three strategies: move-optimal, simple greedy and smart greedy. The move-optimal strategy is a global optimal deployment method that is mainly done by the leading agent. In this method, we demonstrated the optimal path to each target in all possible double loop cases. For the simple greedy and smart greedy strategies, we described two local approaches that allow the surrounding agents to decide their next step according to local information. These two strategies are not optimal.

In chapter 5, we described our solution in triple loop chordal rings. In order to disinfect the entire topology we require a maximum of 24 agents. The maximum number of black viruses originated is six. For the deployment phase we only described the move-optimal strategy since the greedy approach does not work for triple loops. We were unable to calculate the optimal paths for triple loops, however, we provided the upper bounds for the optimal loops. In order to get a better understanding of the problem, we studied two extreme triple loop cases and were able to find the optimal complexity for both cases. For simplicity, we only considered the shortly-chorded triple loop for the analysis of the surrounding phase.

In chapter 6, we addressed the black virus disinfection problem in consecutive-chords loops. In order to disinfect the entire topology we require a maximum of $4k+2$ agents. The maximum number of black viruses originated is $2k$, where k represents the longest chord. In the second phase we described a local strategy, the one-direction greedy strategy, which gives us the shortest paths to the targets.

In chapter 7, we discussed the black virus disinfection problem in general chordal rings. In order to disinfect a chordal ring $C(d_1 = 1, \ldots, d_m)$ we require $O(m^2)$ agents. The maximum number of black viruses originated is $2m$, where m represents the number of chords in one direction. For the deployment phase we described a general protocol that is not optimal but that works correctly for any chord structure.

The following table summarizes the worst case complexities for the various chord structures considered in this thesis. The indicated complexity of the second phase corresponds with the best technique in terms of the number of moves required.

	$C(1,k)$	$C(1,p,k)$	$C(1,2,\ldots,k)$	$C(d_1,d_2,\ldots,d_m)$
Agents	12	24	$4k+2$	$O(m^2)$
Spread	4	6	$2k$	$2m$
Moves Phase1	$O(n)$	$O(n)$	$O(nk)$	$O(nm)$
Moves Phase 2	≤ 36	≤ 78	$O(k)$	$O(m^2)$

Table 8.1: A summary of the worst case complexities for various chordal ring types.

As previously mentioned, the problem of disinfecting a topology from black viruses is fairly new, and therefore, there remain many problems to be solved:

- In the case of directed chordal rings, how can the agents safely explore in an asynchronous environment? In addition, can the routing always be performed correctly avoiding the black viruses?
- In the case of sequential activation, where the triggering of black viruses is done sequentially, what would be the effect on the number of agents and the number of moves?
- In our study we only considered situations in which there is initially a single black virus. What would the topological impact be if we started with an unknown number of black viruses? In consecutive-chord loops, having two black viruses disconnects the graph. What about other types of chordal rings? What would the minimum number of black viruses that would cause network disconnection?

References

[1] L. Barrière. Symmetry properties of chordal rings of degree 3. *Discrete applied mathematics*, pages 211–232, 2003.

[2] L. Barrière, J. Fàbrega, E. Simó, and M. Zaragozá. Fault-tolerant routings in chordal ring networks. *Networks*, pages 180–190, 2000.

[3] L. Barrière, P. Flocchini, P. Fraigniaud, and N. Santoro. Capture of an intruder by mobile agents. In *Proceedings of the Fourteenth Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 200–209, 2002.

[4] R Breisch. An intuitive approach to speleotopology. *Southwestern cavers*, pages 72–78, 1967.

[5] J. Cai, P. Flocchini, and N. Santoro. Decontaminating a network from a black virus. *International Journal of Networking and Computiing*, pages 151–173, 2014.

[6] J. Cai, G. Havas, B. Mans, A. Nerurkar, J. Seifert, and I. Shparlinski. On routing in circulant graphs. In *Computing and Combinatorics*, pages 360–369. Springer, 1999.

[7] J. Chalopin, S. Das, A. Labourel, and E. Markou. Tight bounds for black hole search with scattered agents in synchronous rings. *Theoretical Computer Science*, pages 70–85, 2013.

[8] C. Cooper and R. Tomasz. Searching for black-hole faults in a network using multiple

agents. In *In Proceeding of 10th International Conference on Principles of Distributed Systems (OPODIS*, pages 320–332, 2006.

[9] Colin Cooper, Ralf Klasing, and Tomasz Radzik. Locating and repairing faults in a network with mobile agents. *Theoretical Computer Science*, pages 1638–1647, 2010.

[10] J. Czyzowicz, S. Dobrev, R. Královič, S. Miklík, and D. Pardubská. Black hole search in directed graphs. In *Proceedings of the 17th International Colloquium on Structural Information and Communication Complexity*, pages 182–194, 2010.

[11] J. Czyzowicz, D. R. Kowalski, E. Markou, and A. Pelc. Searching for a black hole in tree networks. In *Proceedings of the 8th International Conference on Principles of Distributed Systems*, pages 67–80, 2004.

[12] S. Dobrev, P. Flocchini, R. Královič, P. Ruzicka, G. Prencipe, and N. Santoro. Black hole search in common interconnection networks. *Networks*, pages 61–71, 2006.

[13] S. Dobrev, P. Flocchini, R. Královič, and N. Santoro. Exploring an unknown graph to locate a black hole using tokens. In *Proceedings of the 4th IFIP International Conference on Theoretical Computer Science*, pages 131–150, 2006.

[14] S. Dobrev, P. Flocchini, G. Prencipe, and N. Santoro. Searching for a black hole in arbitrary networks: Optimal mobile agents protocols. *Distributed Computing*, pages 153–162, 2006.

[15] S. Dobrev, P. Flocchini, G. Prencipe, and N. Santoro. Mobile search for a black hole in an anonymous ring. *Algorithmica*, pages 67–90, 2007.

[16] P. Flocchini. Contamination and decontamination in majority-based systems. *Journal of Cellular Automata*, pages 183 – 200, 2009.

[17] P. Flocchini, F. Geurts, and N. Santoro. Optimal irreversible dynamos in chordal rings. *Discrete Applied Mathematics*, pages 23–42, 2001.

[18] P. Flocchini, M. Huang, and F. Luccio. Decontamination of chordal rings and tori. In *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, pages 8–pp, 2006.

[19] P. Flocchini, M. Huang, and F. Luccio. Decontamination of hypercubes by mobile agents. *Networks*, pages 167–178, 2008.

[20] P. Flocchini, D. Ilcinkas, and N. Santoro. Ping pong in dangerous graphs: Optimal black hole search with pebbles. *Algorithmica*, pages 1006–1033, 2012.

[21] P. Flocchini, M. Kellett, P. Mason, and N. Santoro. Map construction and exploration by mobile agents scattered in a dangerous network. In *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pages 1–10, 2009.

[22] P. Flocchini, M. Kellett, P. C. Mason, and N. Santoro. Searching for black holes in subways. *Theory of Computing Systems*, pages 158–184, 2012.

[23] P. Flocchini, F. Luccio, and L. Song. Size Optimal Strategies for Capturing an Intruder in Mesh Networks. In *Communications in Computing*, pages 200–206, 2005.

[24] P. Flocchini, B. Mans, and N. Santoro. Tree decontamination with temporary immunity. In *Proceedings of the 19th International Symposium on Algorithms and Computation*, pages 330–341, 2008.

[25] P. Flocchini, A. Nayak, and M. Xie. Enhancing peer-to-peer systems through redundancy. *IEEE Journal on Selected Areas in Communications*, pages 15–24, 2007.

[26] S. Kim and T. Robertazzi. Modeling mobile agent behavior. *Computers & Mathematics with Applications*, pages 951 – 966, 2006.

[27] A. Kosowski, A. Navarra, and C. Pinotti. Synchronization helps robots to detect black holes in directed graphs. In *Principles of Distributed Systems*, pages 86–98. Springer, 2009.

[28] A. Kosowski, A. Navarra, and C. M. Pinotti. Synchronous black hole search in directed graphs. *Theoretical Computer Science*, pages 5752–5759, 2011.

[29] F. Luccio, L. Pagli, and N. Santoro. Network decontamination in presence of local immunity. *International Journal of Foundations of Computer Science*, pages 457–474, 2007.

[30] B. Mans. On the interval routing of chordal rings. In *Parallel Architectures, Algorithms, and Networks, 1999.(I-SPAN'99) Proceedings. Fourth InternationalSymposium on*, pages 16–21, 1999.

[31] B. Mans and N. Santoro. Optimal fault-tolerant leader election in chordal rings. In *Fault-Tolerant Computing, 1994. FTCS-24. Digest of Papers., Twenty-Fourth International Symposium on*, pages 392–401, 1994.

[32] A. Nayak, L. Pagli, and N. Santoro. Efficient construction of catastrophic patterns for vlsi reconfigurable arrays. *Integration, the VLSI journal*, pages 133–150, 1993.

[33] A. Nayak, J. Ren, and N. Santoro. An improved testing scheme for catastrophic fault patterns. *Information processing letters*, pages 199–206, 2000.

[34] A. Nayak, N. Santoro, and R. Tan. Fault-intolerance of reconfigurable systolic arrays. In *Fault-Tolerant Computing, 1990. FTCS-20. Digest of Papers., 20th International Symposium*, pages 202–209, 1990.

[35] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. *Networking, IEEE/ACM Transactions on*, pages 17–32, 2003.

APPENDICES

Appendix A

Detailed Paths Analysis in Triple Loops

Now let us consider the different routes to each target in 6 cases depending on the location of the *black virus*. Let $\pi[x_0, x_i]$ denote a path to reach target x_i, and let $dif = k - p$, we have the following situations:

- **Case1**: Let us study the case of finding the *black virus* in the third segment of the chordal ring: $k \leq |S_{area}| < n - k$. In this case, triggering the original *black virus* creates three more *black viruses* : x_1, x_p and x_k, and thus $\mathcal{T}$=$\{x_2$, x_{p-1}, x_{p-k}, x_{k-1}, x_{p+1}, x_{k+1}, x_{k-p}, x_{k+p}, x_{2p}, $x_{2k}\}$.

  - x_{k-1}: Node x_{k-1} is reached through σ_{k-1}

$$\sigma_{k-1} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k} x_{k-1}$$

  - x_{p-1}: Node x_{k-1} is reached through σ_{p-1}

$$\sigma_{p-1} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+p} x_{p-1}$$

- x_2: could be reached in different ways:

$$\pi[x_0, x_2] = \min\{\pi_1, \pi_2, \pi_3, \}$$

  * taking advantage of the fact that x_{-k} is known to be safe:

$$\pi_1 = x_0 \xrightarrow{-k} x_{-k} \xrightarrow{+1} x_{1-k} \xrightarrow{+1} x_{2-k} \xrightarrow{+k} x_2$$

  * taking advantage of the fact that x_{-p} is known to be safe:

$$\pi_2 = x_0 \xrightarrow{-p} x_{-p} \xrightarrow{+1} x_{1-p} \xrightarrow{+1} x_{2-p} \xrightarrow{+p} x_2$$

  * If $p = 4$

$$\pi_3 = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+p} x_{p-1} \xrightarrow{-1} x_2$$

  * If $p = 3$

$$\pi_3 = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+p} x_2$$

- x $_{p+1}$:

  * Taking advantage of the fact that x_{-k} is known to be safe:

$$\pi_4 = x_0 \xrightarrow{-k} x_{-k} \xrightarrow{+p} x_{-k+p} \xrightarrow{+1} x_{-k+p+1} \xrightarrow{+k} x_{p+1}$$

  * If $k = 2p$

$$\pi_5 = x_0 \xrightarrow{-p} x_{-p} \xrightarrow{+1} x_{-k+p+1} \xrightarrow{+k} x_{p+1}$$

  * If $k = 2p + 1$

$$\pi_5 = x_0 \xrightarrow{-p} x_{-k+p+1} \xrightarrow{+k} x_{p+1}$$

- x$_{k-p}$:

$$\pi[x_0, x_{k-p}] = \min\{\pi_6, \sigma_{k-p}\}$$

where

$$\sigma_{k-p} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k} x_{k-1} \xrightarrow{-p} x_{k-p-1} \xrightarrow{+1} x_{k-p}$$

$\pi_6 =$

* If $k < 2p$

$$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+p} x_{p-1} \xrightarrow{-1} x_{p-2}... \xrightarrow{-1} x_{k-p}$$

* If $p = dif$, there is no x$_{k-p}$.
* Else, x$_{k-p}$ is between x$_p$ and x$_k$.

$$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k} x_{k-1} \xrightarrow{-1} x_{k-2}... \xrightarrow{-1} x_{k-p}$$

If applicable(i.e., $k \neq 2p+1$)

- x_{2p} is reached through σ_{2p}:

$$\sigma_{2p} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+p} x_{p-1} \xrightarrow{+p} x_{2p-1} \xrightarrow{+1} x_{2p}$$

- x_{2k} is reached through σ_{2k}:

$$\sigma_{2k} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k} x_{k-1} \xrightarrow{+k} x_{2k-1} \xrightarrow{+1} x_{2k}$$

- x_{k+1}:

$$\pi[x_0, x_{k+1}] = \min\{\pi_7, \sigma_{k+1}, \pi_8, \pi_9\}$$

where

$$\sigma_{k+1} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k} x_{k-1} \xrightarrow{+k} x_{2k-1} \xrightarrow{+1} x_{2k} \xrightarrow{+1} x_{2k+1} \xrightarrow{-k} x_{k+1}$$

$\pi_7 =$

- * If $k < 2p$

$$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+p} x_{p-1} \xrightarrow{+p} x_{2p-1} \xrightarrow{-1} x_{2p-2} \xrightarrow{-1}, ..., \xrightarrow{-1} x_{k+1}$$

- * If $k > 2p$

$$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+p} x_{p-1} \xrightarrow{+k} x_{k+p-1} \xrightarrow{-i} x_{k+p-i}, ..., x_{k+1}$$

  where $i = 1$ or $i = p$ depending on whether the differnece between x_{k+1} and x_{k+p-1} is greater than p or not.

If $|\pi[x_0, x_{k-p}]| < 4$, then x$_{k+1}$ is reached through π_6 plus two more moves.

$$\pi_8 = \pi_6 + x_{k-p} \xrightarrow{+1} x_{k-p+1} \xrightarrow{+p} x_{k+1}$$

If $|\pi[x_0, x_2]| < 4$, then x$_{k+1}$ is reached through π_3 plus two more moves.

$$\pi_9 = \pi_3 + x_2 \xrightarrow{+k} x_{k+2} \xrightarrow{-1} x_{k+1}$$

– x_{k+p} is reached through σ_{k+p}:

$$\sigma_{k+p} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+p} x_{p-1} \xrightarrow{+k} x_{k+p-1} \xrightarrow{+1} x_{k+p}$$

– x_{p-k}

$$\pi[x_0, x_{p-k}] = \min\{\pi_{10}, \sigma_{p-k}\}$$

where

$$\sigma_{k+p} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+p} x_{p-1} \xrightarrow{-k} x_{p-k-1} \xrightarrow{+1} x_{p-k}$$

$\pi_{10} =$

    * If $dif \leq 3$

$$x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-1} x_{-2}, ..., \xrightarrow{-1} x_{p-k}$$

    * If $p \leq 3$

$$x_0 \xrightarrow{-k} x_{-k} \xrightarrow{+1} x_{-k+1} \xrightarrow{+1}, ..., \xrightarrow{+1} x_{p-k}$$

    * If $p - dif < 3$

$$x_0 \xrightarrow{-p} x_{-p} \xrightarrow{+1} x_{-p+1} \xrightarrow{+1}, ..., \xrightarrow{+1} x_{p-k}$$

  - Nodes x_{-p+1}, and x_{-k+1} are occupied by SHs that were at nodes x_{-p} and x_{-k} respectively when the original *black virus* got triggered..

  We have to take into consideration the fact that some of the above paths might be not applicable if they pass thorough a BV. However, the special paths are always applicable.

- **Case2**: In the case of finding the *black virus* in the fourth segment $n - k \leq |S_{area}| < n-p$, two *black viruses* are generated, which are at x_1 and x_p, since the rest neighbours are explored and guarded. Thus, $\mathcal{T}$=$\{x_2, x_{p-1}, x_{p+1}x_{p-k}x_{p+k}x_{2p}\}$

  - x_2 is reached using π_1 or π_2 orπ_3 as we discussed in **Case1**.
  - x_{p-1} is reached using σ_{p-1}
  - x_{p+1} is reached using π_4 or π_5 as we discussed in **Case1**.
  - x_{p-k} is reached using σ_{p-k} or π_{10} as we discussed in **Case1**.
  - x_{p+k} is reached using σ_{p+k}.
  - x_{2p} is reached using σ_{2p}.
  - x_{k+1}, x_{-p+1}, and x_{-k+1} are occupied by SHs that were at nodes x_k, x_{-p} and x_{-k} respectively when the original *black virus* got triggered.

- **Case3**: In the case of finding the *black virus* in the fifth segment $n - p \leq |S_{area}| <$

$n-1$, one *black virus* is generated, which is at x_1 since the rest neighbours are explored and guarded. Thus, $\mathcal{T}$=$\{x_2\}$

- x_2 is reached using π_1 or π_2 orπ_3 as we discussed in **Case1**.
- Nodes x_{p+1}, x_{k+1}, x_{-p+1}, and x_{-k+1} are occupied by SHs that were at nodes x_p, x_k, x_{-p} and x_{-k} respectively when the original *black virus* got triggered.

- **Case4**: We have a special case when the *black virus* is located at node $n-1$. In this case, all neighbours are guarded and no more *black viruses* are created. No more moves are made in the second phase since all moves are done in the first phase as we explained in the previous section.

- **Case5**: In the case of finding the *black virus* in the second segment $p \leq |S_{area}| < k$, four *black viruses* are generated, which are at $\mathcal{BV}$=$\{x_1, x_p, x_k, x_{-k}\}$ since only one SH has been deployed so far at node x_{-p}. Thus, $\mathcal{T}$=$\{x_2$, x_{p-1}, x_{k-1}, x_{p+1}, x_{k+1}, x_{k-p}, x_{k+p}, x_{2p}, x_{2k}, x_{-k+p}, x_{-k+1}, x_{-k-1}, x_{-k-p}, $x_{-2k}\}$. To reach any of these targets, we should avoid any path that has x_{-k}.

  - Node x_2 is reached as the following:

$$\pi[x_0, x_2] = \min\{\pi_2, \pi_3\}$$

  - Node x_{p-1} is reached using σ_{p-1}
  - Node x_{k-1} is reached using σ_{k-1}
  - Node x_{p+1} is reached as the following:

$$\pi[x_0, x_{p+1}] = \min\{\pi_5, \sigma_{p+1}\}$$

- Node x_{k+1} is reached as the following:

$$\pi[x_0, x_{k+1}] = \min\{\pi_7, \pi_8, \pi_9, \sigma_{k+1}\}$$

- Node x_{k-p} is reached as the following:

$$\pi[x_0, x_{k-p}] = \min\{\pi_6, \sigma_{k-p}\}$$

- Node x_{k+p} is reached using σ_{k+p}
- Node x_{2p} is reached using σ_{2p}
- Node x_{2k} is reached using σ_{2k}
- Node x_{-k+p} is reached as the following:

$$\pi[x_0, x_{-k+p}] = \min\{\pi_{10}, \sigma_{-k+p}\}$$

- Node x_{-k+1} is reached as the following:

$$\pi[x_0, x_{-k+1}] = \min\{\pi_{11}, \sigma_{-k+1}\}$$

  where

$$\pi_{11} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-p} x_{-p-1} \xrightarrow{-i}, \ldots \xrightarrow{-i}, x_{-k+1}$$

  where $i = 1$ or $i = p$ depending on the distance $dist$ between x_{-p-1} and x_{-k+1}. If $dist \geq p$, then $i = p$, otherwise, $i = 1$.

- Node x_{-k-1} is reached using σ_{-k-1}
- Node x_{-k-p} is reached using σ_{-k-p}
- Node x_{-2k} is reached using σ_{-2k}

  - Node x_{-p+1} is already guarded by a SH that was at node x_{-p} when the original *black virus* got triggered.

- **Case6**. Finding the *black virus* in the first segment $1 \leq |S_{area}| < p$, five *black viruses* are generated, which are at $\mathcal{BV}$=$\{x_1, x_p, x_k, x_{-p}, x_{-k}\}$ since no SHs have been deployed yet-p. Thus, $\mathcal{T}$=$\{x_2$, x_{p-1}, x_{k-1}, x_{p+1}, x_{k+1}, x_{k-p}, x_{k+p}, x_{2p}, x_{2k}, x_{-p+1}, x_{-p-1}, x_{-2p},x_{-k+p}, x_{-k+1}, x_{-k-1}, x_{-k-p}, $x_{-2k}\}$. To reach any of these targets, we should avoid any path that has x_{-p} or x_{-k} as the following:

  - Node x_2. Since x_{-p} and x_{-k} are BVs, we have

$$\pi[x_0, x_2] = \min\{\pi_{12}, \sigma_2\}$$

where

$$\pi_{12} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+p} x_{p-1} \xrightarrow{-1} x_{p-2} ... \xrightarrow{-1} x_2$$

$$\sigma_2 = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{+k} x_{k-1} \xrightarrow{+k} x_{2k-1} \xrightarrow{+1} x_{2k} \xrightarrow{+1} x_{2k+1} \xrightarrow{+1} x_{2k+2} \xrightarrow{-k} x_{k+2} \xrightarrow{-k} x_2$$

  - Node x_{p-1} is reached using σ_{p-1}
  - Node x_{k-1} is reached using σ_{k-1}
  - Node x_{p+1} is reached using σ_{p+1}
  - Node x_{k+1} is reached as the following:

$$\pi[x_0, x_{k+1}] = \min\{\pi_7, \pi_8, \pi_9, \sigma_{k+1}\}$$

- Node x_{k-p} is reached as the following:

$$\pi[x_0, x_{k-p}] = \min\{\pi_6, \sigma_{k-p}\}$$

- Node x_{k+p} is reached using σ_{k+p}
- Node x_{2p} is reached using σ_{2p}
- Node x_{2k} is reached using σ_{2k}
- Node x_{-p+1} is reached using σ_{-p+1}
- Node x_{-p-1} is reached using σ_{-p-1}
- Node x_{-2p} is reached using σ_{-2p}
- Node x_{p-k} is reached as the following:

$$\pi[x_0, x_{p-k}] = \min\{\pi_{10}, \sigma_{p-k}\}$$

where

$$\pi_{10} = x_0 \xrightarrow{-1} x_{-1} \xrightarrow{-1} x_{-2}, ..., \xrightarrow{-1} x_{p-k}$$

- Node x_{-k+1} is reached as the following:

$$\pi[x_0, x_{-k+1}] = \min\{\pi_{11}, \sigma_{-k+1}\}$$

- Node x_{-k-1} is reached using σ_{-k-1}
- Node x_{-k-p} is reached using σ_{-k-p}
- Node x_{-2k} is reached using σ_{-2k}